Atty. Dkt. No. 040405-0368

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Atsufumi SHIBAYAMA et al.

Title: PROGRAM PARALLELIZATION DEVICE, PROGRAM PARALLELIZATION METHOD, AND PROGRAM PARALLELIZATION PROGRAM

Appl. No.: Unassigned

Filing Date: March 30, 2004

Examiner: Unassigned

Art Unit: Unassigned

CLAIM FOR CONVENTION PRIORITY

Commissioner for Patents
PO Box 1450
Alexandria, Virginia 22313-1450

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested, and the right of priority provided in 35 U.S.C. § 119 is hereby claimed.

In support of this claim, filed herewith is a certified copy of said original foreign application:

Japanese Patent Application No. 2003-093076
filed 03/31/2003.

Respectfully submitted,

Date: March 30, 2004

By _____ Reg No. 38,819
for / David A. Blumenthal
Attorney for Applicant
Registration No. 26,257

FOLEY & LARDNER LLP
Customer Number: 22428
Telephone: (202) 672-5407
Facsimile: (202) 672-5399

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 Date of Application:	２００３年　３月３１日
出願番号 Application Number: [ST. 10/C]:	特願２００３－０９３０７６ [ＪＰ２００３－０９３０７６]
出願人 Applicant(s):	日本電気株式会社

２００３年１０月１７日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

出証番号　出証特２００３－３０８５５４５

【書類名】　特許願
【整理番号】　34002301
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/46
　　G06F 15/16
【発明者】
　【住所又は居所】　東京都港区芝五丁目７番１号　日本電気株式会社内
　【氏名】　柴山　充文
【発明者】
　【住所又は居所】　東京都港区芝五丁目７番１号　日本電気株式会社内
　【氏名】　大澤　拓
【発明者】
　【住所又は居所】　東京都港区芝五丁目７番１号　日本電気株式会社内
　【氏名】　松下　智
【特許出願人】
　【識別番号】　000004237
　【氏名又は名称】　日本電気株式会社
【代理人】
　【識別番号】　100088890
　【弁理士】
　【氏名又は名称】　河原　純一
【手数料の表示】
　【予納台帳番号】　009690
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　9001717

【プルーフの要否】　　要

【書類名】　　明細書

【発明の名称】プログラム並列化装置，プログラム並列化方法およびプログラム並列化プログラム

【特許請求の範囲】

【請求項１】逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部と、

前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部と、

与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部と、

前記フォーク箇所候補決定部により決定されたフォーク箇所候補の試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部と、

前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部と

を有することを特徴とするプログラム並列化装置。

【請求項２】逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部と、

前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部と、

与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部と、

前記フォーク箇所候補決定部により決定されたフォーク箇所候補からフォーク１

回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部と、

前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部と

を有することを特徴とするプログラム並列化装置。

【請求項３】前記並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とする請求項１または請求項２記載のプログラム並列化装置。

【請求項４】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とする請求項１，請求項２または請求項３記載のプログラム並列化装置。

【請求項５】前記フォーク箇所候補最良組合せ決定部が、求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を前記並列実行性能評価部によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとすることを特徴とする請求項４記載のプログラム並列化装置。

【請求項６】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようにフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実

行性能を前記並列実行性能評価部によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とする請求項１，請求項２または請求項３記載のプログラム並列化装置。

【請求項７】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部の処理が終了すると、前記フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに２つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とする請求項６記載のプログラム並列化装置。

【請求項８】逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法のためのプログラム並列化方法において、

逐次処理プログラムの制御フローおよびデータフローの解析を行う工程と、

制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定する工程と、
決定されたフォーク箇所候補からフォーク箇所候補の試行組合せを生成する工程と、
生成されたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する工程と、
評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定する工程と、
決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程と
を含むことを特徴とするプログラム並列化方法。
【請求項９】逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法のためのプログラム並列化方法において、
逐次処理プログラムの制御フローおよびデータフローの解析を行う工程と、
制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定する工程と、
決定されたフォーク箇所候補からフォーク１回モデル上で互いに同時に実行可能なフォーク箇所候補のみからなる試行組合せを生成する工程と、
生成されたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する工程と、
評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定する工程と、
決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程と
を含むことを特徴とするプログラム並列化方法。
【請求項１０】制御・データフロー解析部が、逐次処理プログラムの制御フローおよびデータフローを解析する工程と、
フォーク箇所候補決定部が、制御・データフロー解析部による制御フローおよび

データフローの解析結果を参照してフォーク箇所候補を生成する工程と、
フォーク箇所候補最良組合せ決定部が、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける工程と、
フォーク箇所候補最良組合せ決定部が、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする工程と、
並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成する工程と、
並列実行性能評価部が、逐次実行トレースをすべてのターム点候補を分割点として分割する工程と、
並列実行性能評価部が、各スレッド要素毎のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する工程と、
フォーク箇所候補最良組合せ決定部が、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を１つ選択する工程と、
フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する工程と、
フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする工程と、
フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補の最良組合せから選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする工程と、
フォーク箇所候補最良組合せ決定部が、試行組合せによる並列化の並列実行性能を並列実行性能評価部によって評価する工程と、
フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能の方が良ければ、試行組合せを現時点でのフォーク箇所候補の最良組合せとする工程と、
フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補があるかどうかを判定し、未選択のフォーク箇所候補があれば繰り返し実行する工程と、
フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補がなければ、直前の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する工程と、
フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つかった場合には、すべてのフォーク箇所候補を未選択の状態にして繰り返し実行する工程と、
フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つからなかった場合には、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として出力する工程と、
並列化プログラム出力部が、フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程と
を含むことを特徴とするプログラム並列化方法。
【請求項１１】制御・データフロー解析部が、逐次処理プログラムの制御フローおよびデータフローを解析する工程と、
フォーク箇所候補決定部が、制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照してフォーク箇所候補を生成する工程と、
フォーク箇所候補最良組合せ決定部が、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける工程と、
フォーク箇所候補最良組合せ決定部が、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする工程と、
並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成する工程と、

並列実行性能評価部が、逐次実行トレースをすべてのターム点候補を分割点として分割する工程と、

並列実行性能評価部が、各スレッド要素毎のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する工程と、

フォーク箇所候補最良組合せ決定部が、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を１つ選択する工程と、

フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する工程と、

フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする工程と、

フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補をキャンセルするフォーク箇所候補を試行組合せから取り除く工程と、

フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補の最良組合せから選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せによる並列化の並列実行性能を並列実行性能評価部によって評価する工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能の方が良ければ、選択されたフォーク箇所候補がキャンセルするフォーク箇所候補を試行組合せから取り除く工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せを現時点でのフォーク箇所候補の最良組合せとする工程と、

フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補があるかどうかを判定し、未選択のフォーク箇所候補があれば繰り返し実行する工程と、

フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補がなければ、直前の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する工程と、

フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つかった場合には、すべてのフォーク箇所候補を未選択の状態にして繰り返し実行する工程と、

フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つからなかった場合には、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として並列化プログラム出力部に出力する工程と、

並列化プログラム出力部が、フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程と

を含むことを特徴とするプログラム並列化方法。

【請求項１２】前記並列実行性能を評価する工程で、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とする請求項８，請求項９，請求項１０または請求項１１記載のプログラム並列化方法。

【請求項１３】前記フォーク箇所候補の最良組合せを決定する工程で、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とする請求項８，請求項９，請求項１０または請求項１１記載のプログラム並列化方法。

【請求項１４】前記フォーク箇所候補の最良組合せを決定する工程で、全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようにフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク

箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能をある入力データに対して評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とする請求項８，請求項９，請求項１０または請求項１１記載のプログラム並列化方法。

【請求項１５】前記フォーク箇所候補の最良組合せを決定する工程で、前記フォーク箇所候補決定部の処理が終了すると、フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに２つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とする請求項８，請求項９，請求項１０または請求項１１記載のプログラム並列化方法。

【請求項１６】コンピュータを、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部，前記制御・データフロー解析部

による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部，与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部，前記フォーク箇所候補決定部により決定されたフォーク箇所候補の試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部，ならびに前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部として動作させることを特徴とするプログラム並列化プログラム。

【請求項１７】コンピュータを、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部，前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部，与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部，前記フォーク箇所候補決定部により決定されたフォーク箇所候補からフォーク１回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部，ならびに前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部として動作させることを特徴とするプログラム並列化プログラム。

【請求項１８】前記並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素

の単位でシミュレートして並列実行性能を算出することを特徴とする請求項１６または請求項１７記載のプログラム並列化プログラム。

【請求項１９】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とする請求項１６，請求項１７または請求項１８記載のプログラム並列化プログラム。

【請求項２０】前記フォーク箇所候補最良組合せ決定部が、求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を前記並列実行性能評価部によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとすることを特徴とする請求項１９記載のプログラム並列化プログラム。

【請求項２１】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようにフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を前記並列実行性能評価部によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とする請求項１６，請求項１７または請求項１８記載のプログラム並列化プログラム。

【請求項２２】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部の処理が終了すると、前記フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行

わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに２つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とする請求項２１記載のプログラム並列化プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、複数のスレッドを複数のプロセッサにより並列に実行するマルチスレッド型プロセッサのためのプログラム変換技術に関し、特に実行効率の高い並列化プログラムを高速に生成するためのプログラム並列化装置，プログラム並列化方法およびプログラム並列化プログラム（コンパイラ）に関する。

【０００２】

【従来の技術】

単一の逐次処理プログラムを並列プロセッサシステムで並列に処理する手法として、プログラムをスレッドと呼ぶ命令流に分割して複数のプロセッサで並列に実行するマルチスレッド実行方法がある（例えば、特許文献１～３，非特許文献１参照）。以下、これらの従来のマルチスレッド実行方法について説明する。

【０００３】

一般に、マルチスレッド実行方法において、他のプロセッサ上に新たなスレッド

を生成することを、スレッドをフォーク（ｆｏｒｋ）するといい、フォーク動作を行った側のスレッドを親スレッド、生成された新しいスレッドを子スレッド、スレッドをフォークするプログラム位置をフォーク点、子スレッドの先頭のプログラム位置をフォーク先アドレスまたは子スレッドの開始点と呼ぶ。特許文献１～３および非特許文献１では、スレッドのフォークを指示するためにフォーク点にフォーク命令が挿入される。フォーク命令にはフォーク先アドレスが指定され、フォーク命令の実行によりそのフォーク先アドレスから始まる子スレッドが他プロセッサ上に生成され、子スレッドの実行が開始される。また、スレッドの処理を終了させるプログラム位置をターム点と呼び、各プロセッサはスレッドの処理を終了する。

【０００４】

図１５は、マルチスレッド実行方法の処理の概要を示す図である。同図（ａ）は、３つのスレッドＡ，Ｂ，Ｃに分割された逐次処理プログラムを示す。このプログラムを単一のプロセッサで処理する場合、同図（ｂ）に示すように１つのプロセッサＰＥがスレッドＡ，Ｂ，Ｃを順番に処理していく。これに対して、特許文献１～３および非特許文献１のマルチスレッド実行方法では、同図（ｃ）に示すように、１つのプロセッサＰＥ１にスレッドＡを実行させ、プロセッサＰＥ１でスレッドＡを実行している最中に、スレッドＡに埋め込まれたフォーク命令によってスレッドＢを他のプロセッサＰＥ２に生成し、プロセッサＰＥ２においてスレッドＢを実行させる。また、プロセッサＰＥ２は、スレッドＢに埋め込まれたフォーク命令によってスレッドＣをプロセッサＰＥ３に生成する。プロセッサＰＥ１，ＰＥ２は、それぞれスレッドＢ，Ｃの開始点の直前のターム点においてスレッドの処理を終了し、プロセッサＰＥ３は、スレッドＣの最後の命令を実行すると、その次の命令（一般にはシステムコール命令）を実行する。このように、複数のプロセッサでスレッドを同時に並行して実行することにより、逐次処理に比べて性能の向上が図られる。

【０００５】

他のマルチスレッド実行方法として、図１５（ｄ）に示すように、スレッドＡを実行しているプロセッサＰＥ１からフォークを複数回行うことにより、プロセッ

サＰＥ２にスレッドＢを、またプロセッサＰＥ３にスレッドＣをそれぞれ生成するマルチスレッド実行方法も存在する。図１５（ｄ）のモデルに対して、同図（ｃ）に示したように、スレッドはその生存中に高々１回に限って有効な子スレッドを生成することができるという制約を課したマルチスレッド実行方法を、フォーク１回モデルと呼ぶ。フォーク１回モデルでは、スレッド管理の大幅な簡略化が可能となり、現実的なハードウェア規模でスレッド管理部のハードウェア化が実現できる。また、個々のプロセッサは子スレッドを生成する他プロセッサが１プロセッサに限定されるため、隣接するプロセッサを単方向にリング状に接続した並列プロセッサシステムでマルチスレッド実行が可能となる。

【０００６】

親スレッドが子スレッドを生成し、子スレッドに所定の処理を行わせるには、親スレッドのフォーク点におけるレジスタファイル中のレジスタのうち少なくとも子スレッドで必要なレジスタの値を親スレッドから子スレッドに引き渡す必要がある。このスレッド間のデータ引き渡しコストを削減するために、非特許文献１および特許文献２では、スレッド生成時のレジスタ値継承機構をハードウェア的に備えている。これは、スレッド生成時に親スレッドのレジスタファイルの内容を子スレッドに全てコピーするものである。子スレッドの生成後は、親スレッドと子スレッドとのレジスタ値の変更は独立となり、レジスタを用いたスレッド間のデータの引き渡しは行われない。スレッド間のデータ引き渡しに関する他の従来技術としては、レジスタの値を命令によりレジスタ単位で個別に転送する機構を備えた並列プロセッサシステムも提案されている。

【０００７】

マルチスレッド実行方法では、実行の確定した先行スレッドを並列に実行することを基本とするが、実際のプログラムでは実行の確定するスレッドが充分に得られない場合も多い。また、動的に決定される依存やコンパイラ解析能力の限界等により並列化率が低く抑えられ、所望の性能が得られない可能性が生じる。このため、特許文献１では、制御投機を導入し、ハードウェア的にスレッドの投機実行をサポートしている。制御投機では、実行する可能性の高いスレッドを実行確定前に投機的に実行する。投機状態のスレッドは、実行の取り消しがハードウェ

ア上可能である範囲内で仮実行を行う。子スレッドが仮実行を行っている状態を仮実行状態といい、子スレッドが仮実行状態にあるとき親スレッドはスレッド仮生成状態にあるという。仮実行状態の子スレッドでは共有メモリおよびキャッシュメモリへの書き込みは抑制され、別途設けた仮実行用バッファ（ｔｅｍｐｏｒａｒｙ
ｂｕｆｆｅｒ）に対して書き込みが行われる。投機が正しいことが確定すると、親スレッドから子スレッドに対して投機成功通知が出され、子スレッドは仮実行用バッファの内容を共有メモリおよびキャッシュメモリに反映し、仮実行用バッファを用いない通常の状態となる。また、親スレッドは、スレッド仮生成状態からスレッド生成状態となる。他方、投機が失敗したことが確定すると、親スレッドでスレッド破棄命令（ａｂｏｒｔ）が実行され、子スレッド以下の実行がキャンセルされる。また、親スレッドは、スレッド仮生成状態からスレッド未生成状態となり、再び子スレッドの生成が可能になる。つまり、フォーク１回モデルでは、スレッド生成は高々１回に限定されるが、制御投機を行い、投機が失敗した場合には再びフォークが可能となる。この場合においても、有効な子スレッドは高々１つである。

【０００８】

スレッドはその生存中に高々１回に限って有効な子スレッドを生成するというフォーク１回モデルのマルチスレッド実行を実現するために、例えば非特許文献１等では、逐次処理プログラムから並列化プログラムを生成するコンパイルの段階で、全てのスレッドが有効なフォークを１回しか実行しない命令コードになるように制限している。すなわち、フォーク１回制限を並列化プログラム上において静的に保証している。

【０００９】

一方、特許文献３では、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する１つのフォーク命令を親スレッドの実行中に選択することにより、フォーク１回制限をプログラム実行時に保証している。

【００１０】

次に、上述したマルチスレッド実行方法で実行される従来の並列化プログラム生

成装置について説明する。

【００１１】

図１６を参照すると、従来のプログラム並列化装置１０は、逐次処理プログラム１３を入力し、制御・データフロー解析部１１によって逐次処理プログラム１３の制御フローおよびデータフローを解析し、次いでその結果に基づき、フォーク挿入部１２において、基本ブロックあるいは複数の基本ブロックを並列化の単位、すなわちスレッドに分割し、次いで並列化のためのフォーク命令を挿入して、複数のスレッドに分割された並列化プログラム１４を生成し出力する。

【００１２】

フォーク命令の挿入箇所（フォーク箇所）は、制御フローおよびデータフローの解析結果を参照して、より並列実行性能が得られるように決定される。しかし、プログラム実行時にしか判明しないプログラム実行フローやメモリ依存関係などの影響により、静的な解析に基づいたフォーク命令の挿入方法では所望の並列実行性能を得られない場合がある。

【００１３】

これに対し、逐次実行を行ったときの条件分岐確率やデータ依存発生頻度などのプロファイル情報を参照して、フォーク箇所を決定する方法が公知である（例えば、特許文献４参照）。この方法によれば、逐次実行時の動的な情報であるプロファイル情報を利用することで、より適切なフォーク命令の挿入が可能となり、並列実行性能の向上が期待できる。

【００１４】

【特許文献１】

特開１０―２７１０８号公報（第５頁、図１）

【非特許文献１】

「Ｏｎ　Ｃｈｉｐ　Ｍｕｌｔｉｐｒｏｃｅｓｓｏｒ指向　制御並列アーキテクチャＭＵＳＣＡＴの提案」（並列処理シンポジュウムＪＳＰＰ９７論文集、情報処理学会、ｐｐ．２２９－２３６、Ｍａｙ　１９９７）

【特許文献２】

特開１０―７８８８０号公報（第７頁、図２）

【特許文献３】

特開２００３－０２９９８５号公報（第１３頁、図１０）

【特許文献４】

特開２００１－２８２５４９号公報（第１１－１２頁、図２）

【００１５】

【発明が解決しようとする課題】

ところで、上記した従来の技術は、下記の問題点を有している。

【００１６】

第１の問題点は、フォーク箇所を決定する基準にまだ多分に改良の余地があるため、従来の技術をもってしても、所望の並列実行性能を得られない場合があることである。その理由は、フォーク箇所を決定する基準は、望ましくはそのフォーク命令の並列実行性能への寄与度であるが、たとえ逐次実行時のプロファイル情報を利用したとしても、並列実行時の性能を解析的な方法で精度良く予測することは困難であるため、適切にフォーク命令を挿入できない場合があるからである。

【００１７】

並列実行性能を解析的に高精度に予測することを困難にしている要因の一つに、並列実行性能に対するメモリ依存関係の影響がある。並列実行性能に対するメモリ依存関係の影響は、並列化の仕方に応じて複雑に変化するため、逐次実行時のプロファイル情報からメモリ依存関係の情報を得たとしても、その並列実行性能への影響を精度良く評価することは困難である。また、フォーク１回モデルなどの並列化方法上の制約や、プロセッサ間接続方式などのハードウェア構成等も並列実行性能に大きな影響を与える。しかし、同様に、これらの並列実行性能への影響は、並列化の仕方に応じて複雑に変化するため、逐次実行時のプロファイル情報から解析的な方法により精度良く評価することは困難である。

【００１８】

すなわち、制御フローおよびデータフローの解析結果や逐次実行時に得られるプロファイル情報からでは、並列実行性能を精度良く予測することは困難であるため、適切にフォーク命令を挿入できず、所望の並列実行性能を得られない場合が

あるという問題がある。

【００１９】

第２の問題点は、より良い並列実行性能のフォーク箇所を得ようとするにつれて、そのフォーク箇所の決定処理にますます時間がかかるということである。その理由は、フォーク箇所を決定する評価基準を高精度化するにつれて、その評価処理にますます時間がかかるのに加えて、フォーク箇所の数は、通常、例えばプログラムに含まれるループ数などと比較しても多く、従ってその組合せの数は膨大になるからである。

【００２０】

本発明は、このような事情に鑑みて提案されたものであり、その目的は、フォーク箇所候補最良組合せ決定処理を高精度かつ高速に行い、並列実行性能のより高い並列化プログラムを高速に生成するプログラム並列化装置，プログラム並列化方法およびプログラム並列化プログラムを提供することにある。

【００２１】

【課題を解決するための手段】

前記目的を達成するため、本発明のプログラム並列化装置，プログラム並列化方法およびプログラム並列化プログラムは、入力される逐次処理プログラムの制御フローおよびデータフローの解析結果や逐次実行時に得られるプロファイル情報に加えて、並列実行性能を直接評価した結果を基準に、フォーク箇所候補の最良組合せを決定することを特徴とする。

【００２２】

具体的には、第１の発明は、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部と、前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部と、与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部と、前記フォーク箇所候補決定部により決定されたフォーク箇所候補の試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列

実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部と、前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部とを有することを特徴とするプログラム並列化装置である。

【００２３】

また、第２の発明は、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部と、前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部と、与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部と、前記フォーク箇所候補決定部により決定されたフォーク箇所候補からフォーク１回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部と、前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部とを有することを特徴とするプログラム並列化装置である。

【００２４】

さらに、第３の発明は、前記並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とするプログラム並列化装置である。

【００２５】

さらにまた、第４の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フ

ォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とするプログラム並列化装置である。

【００２６】

また、第５の発明は、前記フォーク箇所候補最良組合せ決定部が、求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を前記並列実行性能評価部によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとすることを特徴とするプログラム並列化装置である。

【００２７】

さらに、第６の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようにフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を前記並列実行性能評価部によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とするプログラム並列化装置である。

【００２８】

さらにまた、第７の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部の処理が終了すると、前記フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォー

ク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに2つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とするプログラム並列化装置である。

【００２９】

また、第８の発明は、逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法のためのプログラム並列化方法において、逐次処理プログラムの制御フローおよびデータフローの解析を行う工程と、制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定する工程と、決定されたフォーク箇所候補からフォーク箇所候補の試行組合せを生成する工程と、生成されたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する工程と、評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定する工程と、決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程とを含むことを特徴とするプログラム並列化方法である。

【００３０】

さらに、第９の発明は、逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法のためのプログラム並列化方法において、逐次処理プログラムの制御フローおよびデータフローの解析を行う工程と、制御フローおよびデータフローの解析結果を参照して逐次処理プログラ

ムのフォーク箇所候補を決定する工程と、決定されたフォーク箇所候補からフォーク１回モデル上で互いに同時に実行可能なフォーク箇所候補のみからなる試行組合せを生成する工程と、生成されたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する工程と、評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定する工程と、決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程とを含むことを特徴とするプログラム並列化方法である。

【００３１】

さらにまた、第１０の発明は、制御・データフロー解析部が、逐次処理プログラムの制御フローおよびデータフローを解析する工程と、フォーク箇所候補決定部が、制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照してフォーク箇所候補を生成する工程と、フォーク箇所候補最良組合せ決定部が、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける工程と、フォーク箇所候補最良組合せ決定部が、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする工程と、並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成する工程と、並列実行性能評価部が、逐次実行トレースをすべてのターム点候補を分割点として分割する工程と、並列実行性能評価部が、各スレッド要素毎のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する工程と、フォーク箇所候補最良組合せ決定部が、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を１つ選択する工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最

良組合せに含まれていれば、フォーク箇所候補の最良組合せから選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする工程と、フォーク箇所候補最良組合せ決定部が、試行組合せによる並列化の並列実行性能を並列実行性能評価部によって評価する工程と、フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する工程と、フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能の方が良ければ、試行組合せを現時点でのフォーク箇所候補の最良組合せとする工程と、フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補があるかどうかを判定し、未選択のフォーク箇所候補があれば繰り返し実行する工程と、フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補がなければ、直前の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する工程と、フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つかった場合には、すべてのフォーク箇所候補を未選択の状態にして繰り返し実行する工程と、フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つからなかった場合には、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として出力する工程と、並列化プログラム出力部が、フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程とを含むことを特徴とするプログラム並列化方法である。

【００３２】

また、第１１の発明は、制御・データフロー解析部が、逐次処理プログラムの制御フローおよびデータフローを解析する工程と、フォーク箇所候補決定部が、制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照してフォーク箇所候補を生成する工程と、フォーク箇所候補最良組合せ決定部が、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける工程と、フォーク箇所候補最良組合せ決定部が、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする工程と、並列実行性能評価

部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成する工程と、並列実行性能評価部が、逐次実行トレースをすべてのターム点候補を分割点として分割する工程と、並列実行性能評価部が、各スレッド要素毎のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する工程と、フォーク箇所候補最良組合せ決定部が、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を１つ選択する工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補をキャンセルするフォーク箇所候補を試行組合せから取り除く工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補の最良組合せから選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする工程と、フォーク箇所候補最良組合せ決定部が、試行組合せによる並列化の並列実行性能を並列実行性能評価部によって評価する工程と、フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する工程と、フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能の方が良ければ、選択されたフォーク箇所候補がキャンセルするフォーク箇所候補を試行組合せから取り除く工程と、フォーク箇所候補最良組合せ決定部が、試行組合せを現時点でのフォーク箇所候補の最良組合せとする工程と、フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補があるかどうかを判定し、未選択のフォーク箇所候補があれば繰り返し実行する工程と、フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補がなければ、直前の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する工程と、フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つかった場合には、すべてのフォーク箇所候補を未選択の状態にして繰り返し実行する工

程と、フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つからなかった場合には、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として並列化プログラム出力部に出力する工程と、並列化プログラム出力部が、フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程とを含むことを特徴とするプログラム並列化方法である。

【００３３】

さらに、第１２の発明は、前記並列実行性能を評価する工程で、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とするプログラム並列化方法である。

【００３４】

さらにまた、第１３の発明は、前記フォーク箇所候補の最良組合せを決定する工程で、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とするプログラム並列化方法である。

【００３５】

また、第１４の発明は、前記フォーク箇所候補の最良組合せを決定する工程で、全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようにフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能をある入力データに対して評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とする

プログラム並列化方法である。

【００３６】

さらに、第１５の発明は、前記フォーク箇所候補の最良組合せを決定する工程で、前記フォーク箇所候補決定部の処理が終了すると、フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに２つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とするプログラム並列化方法である。

【００３７】

さらにまた、第１６の発明は、コンピュータを、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部，前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部，与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部，前記

フォーク箇所候補決定部により決定されたフォーク箇所候補の試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部，ならびに前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部として動作させることを特徴とするプログラム並列化プログラムである。

【００３８】

また、第１７の発明は、コンピュータを、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部，前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部，与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部，前記フォーク箇所候補決定部により決定されたフォーク箇所候補からフォーク１回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部，ならびに前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部として動作させることを特徴とするプログラム並列化プログラムである。

【００３９】

さらに、第１８の発明は、前記並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とするプログラム並列化プログラムである。

【００４０】

さらにまた、第１９の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とするプログラム並列化プログラムである。

【００４１】

また、第２０の発明は、前記フォーク箇所候補最良組合せ決定部が、求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を前記並列実行性能評価部によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとすることを特徴とするプログラム並列化プログラムである。

【００４２】

さらに、第２１の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようにフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を前記並列実行性能評価部によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とするプログラム並列化プログラムである。

【００４３】

さらにまた、第２２の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部の処理が終了すると、前記フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割

下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに２つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とするプログラム並列化プログラムである。

【００４４】

本発明によれば、フォーク箇所候補の試行組合せに対して並列実行性能を直接評価し、それを基準にフォーク箇所候補の最良組合せを決定するので、フォーク命令を逐次処理プログラムにより適切に挿入することが可能であり、並列実行性能のより高い並列化プログラムを得ることができる。

【００４５】

また、本発明によれば、フォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくため、フォーク箇所候補のすべての組合せを評価する場合に比べて大幅にフォーク箇所候補最良組合せ決定処理の処理時間を短縮することができるという効果がある。

【００４６】

さらに、本発明によれば、マルチスレッド実行方法としてフォーク１回モデルを採用した場合に、互いに同時に実行可能なフォーク箇所候補の試行組合せのみを

対象に並列実行性能を評価するので、フォーク箇所候補最良組合せ決定処理の処理時間を短縮することができるという効果がある。

【００４７】

さらにまた、本発明によれば、フォーク箇所候補をグループ間では互いになるべく影響を及ぼし合わないように適当数のグループに分割して、各グループ毎に独立にフォーク箇所候補最良組合せ決定処理を行うので、フォーク箇所候補の組合せの総数を削減することができ、フォーク箇所候補の数が多い場合でもフォーク箇所候補最良組合せ決定処理を高速に行うことができるという効果がある。

【００４８】

また、本発明によれば、ターム点候補で決定されるスレッド要素の単位で並列実行のシミュレーションを行うので、並列実行性能を高速に評価することが可能であり、従ってフォーク箇所候補最良組合せ決定処理を高速に行うことができるという効果がある。

【００４９】

図３を参照して具体例を説明する。図３は、隣接するプロセッサを単方向にリング状に接続した並列プロセッサシステム上で、フォーク１回モデルによるマルチスレッド実行の概略を示す図である。同図（ａ）は、３つのスレッドＡｎ，Ｂｎ，Ｃｎ（ｎ＝…，１，２，３，…）が繰り返し実行される逐次処理プログラムの逐次実行トレースを示す。ここで、フォーク箇所候補として、スレッドＡｎの先頭をフォーク点、スレッドＡｎの終端をフォーク先とするフォーク箇所候補ｆ０と、スレッドＢｎの先頭をフォーク点、スレッドＢｎの終端をフォーク先とするフォーク箇所候補ｆ１と、スレッドＢｎの先頭をフォーク点、スレッドＣｎの終端をフォーク先とするフォーク箇所候補ｆ２とがあるとする。

【００５０】

３つのフォーク箇所候補ｆ０，ｆ１，ｆ２の組合せは、ϕ（フォークしない）、｛ｆ０｝，｛ｆ１｝，｛ｆ２｝，｛ｆ０，ｆ１｝，｛ｆ０，ｆ２｝，｛ｆ１，ｆ２｝，｛ｆ０，ｆ１，ｆ２｝の８通り存在する。フォーク箇所候補最良組合せ決定処理は、フォーク箇所候補の組合せの中からより良い並列実行性能を示す組合せを選択する処理に他ならない。仮にいずれのフォーク箇所候補の組合せを選択

してもデータ依存関係を壊さないとすると、図３（ｂ）は｛ｆ０｝により並列実行した（フォーク箇所候補ｆ０のみにフォーク命令を挿入した）ときの様子を示す図である。同様に、同図（ｃ）は｛ｆ１｝により、同図（ｄ）は｛ｆ２｝により、同図（ｅ）は｛ｆ０，ｆ１｝により、同図（ｆ）は｛ｆ０，ｆ２｝により、並列実行したときの様子を示す図である。

【００５１】

いま、フォーク１回モデルを仮定しているので，フォーク箇所候補ｆ１とフォーク箇所候補ｆ２とは同時に実行することができず、｛ｆ１，ｆ２｝および｛ｆ０，ｆ１，ｆ２｝はそのまま並列実行することができない。従って、非特許文献１等で示されているように並列化コードの生成時に高々どちらか一方のみが挿入されるように保証するか、または特許文献３で示されているように実行時にどちらか一方のみが選択されて実行されることになる。

【００５２】

図３（ａ）に示した逐次実行した場合を基準にすると、同図（ｂ）の｛ｆ０｝による並列化は１．３３倍の処理量を提供しているため、逐次実行性能の１．３３倍の並列実行性能を達成している。すなわち、逐次実行に対して１．３３倍プログラム実行が高速である。同様に、同図（ｃ）の｛ｆ１｝による並列化では２．６７倍、同図（ｄ）の｛ｆ２｝による並列化では４倍、同図（ｅ）の｛ｆ０，ｆ１｝による並列化では３．２倍、同図（ｆ）の｛ｆ０，ｆ２｝による並列化では２．６７倍の並列実行性能を達成している。すなわち、本具体例では、例えばフォーク箇所候補ｆ０とフォーク箇所候補ｆ１またはｆ２とを組み合わせるよりも、フォーク箇所候補ｆ２単独とした方がより良い並列実行性能を示す。これは、フォーク１回制限やプロセッサが単方向にリング状に接続されているというハードウェア構成も大きく影響しており、さらにデータ依存関係に違反するような状況ではより複雑な様相を呈する。このため、逐次処理プログラムの制御フローおよびデータフローの解析結果や逐次実行時に得られるプロファイル情報のみからでは、このような並列実行性能を予測することは困難である。従って、従来の並列化方法では所望の並列実行性能を得られない場合があったのに対して、本発明によれば、並列実行性能を直接評価し、それを基準にしてフォーク箇所候補の最

良組合せを決定するので、フォーク命令を逐次処理プログラムにより適切に挿入することが可能であり、並列実行性能のより高い並列化プログラムを得ることができる。

【００５３】

【発明の実施の形態】

以下、本発明の実施の形態について図面を参照して詳細に説明する。

【００５４】

［第１の実施の形態］

図１を参照すると、本発明の第１の実施の形態に係るプログラム並列化装置２０は、入力される逐次処理プログラム１３の制御フローおよびデータフローの解析を行う制御・データフロー解析部２１と、制御・データフロー解析部２１による制御フローおよびデータフローの解析結果を参照して逐次処理プログラム１３のフォーク箇所候補を決定するフォーク箇所候補決定部２２と、フォーク箇所候補決定部２２により決定されたフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を並列実行性能評価部２４によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部２３と、フォーク箇所候補最良組合せ決定部２３より与えられるフォーク箇所候補の試行組合せにより逐次処理プログラム１３を並列化したときの並列実行性能を、ある入力データ１５に対して評価する並列実行性能評価部２４と、フォーク箇所候補最良組合せ決定部２３により決定されたフォーク箇所候補の最良組合せに基づいて並列化プログラム１４を生成し出力する並列化プログラム出力部２５とを備える。

【００５５】

図２を参照すると、フォーク箇所候補決定部２２，フォーク箇所候補最良組合せ決定部２３，および並列実行性能評価部２４によるフォーク箇所候補最良組合せ決定処理は、フォーク箇所候補生成ステップ１００と、フォーク箇所候補順序付けステップ１０１と、フォーク箇所候補初期組合せ生成ステップ１０２と、逐次実行トレース生成ステップ１０３と、スレッド要素分割ステップ１０４と、スレッド要素情報解析・記憶ステップ１０５と、フォーク箇所候補選択ステップ１０

６と、フォーク箇所候補最良組合せ判定ステップ１０７と、フォーク箇所候補加入・試行組合せ設定ステップ１０８と、フォーク箇所候補削除・試行組合せ設定ステップ１０９と、並列実行性能算出ステップ１１０と、並列実行性能比較ステップ１１１と、最良組合せ設定ステップ１１２と、全フォーク箇所候補選択判定ステップ１１３と、新最良組合せ発見判定ステップ１１４と、全フォーク箇所候補未選択設定ステップ１１５と、最良組合せ出力ステップ１１６とからなる。

【００５６】

次に、このように構成された第１の実施の形態に係るプログラム並列化装置２０の動作について説明する。

【００５７】

特に、図２を参照して、フォーク箇所候補決定部２２，フォーク箇所候補最良組合せ決定部２３，および並列実行性能評価部２４で処理されるフォーク箇所候補最良組合せ決定処理を詳細に説明する。

【００５８】

まず、制御・データフロー解析部２１は、逐次処理プログラム１３の制御フローおよびデータフローを解析する。

【００５９】

次に、フォーク箇所候補決定部２２は、制御・データフロー解析部２１による制御フローおよびデータフローの解析結果を参照して、フォーク箇所候補を生成する（ステップ１００）。具体的には、制御フローを解析して逐次処理プログラム１３の基本ブロックを抽出し、その基本ブロックあるいは複数の基本ブロックを単位として並列化を行うようにフォーク箇所候補を生成する。このとき、例えば、非特許文献１や特許文献２に開示されているスレッド生成時にレジスタ値を子スレッドに継承し、子スレッド生成後は親スレッドと子スレッド間でレジスタを用いたデータの引き渡しを行わないマルチスレッド実行方法では、データフロー上でレジスタの内容が同じである任意の箇所でフォークを行うことが可能であり、フォーク箇所候補となり得る。また、非特許文献１（４．１節）に開示されている方針などにより、制御フローおよびデータフローを参照して並列化性能に対して有効だと予測されるフォーク箇所候補をある程度あらかじめ絞り込んでもよ

い。

【００６０】

次に、フォーク箇所候補最良組合せ決定部２３は、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける（ステップ１０１）。効果の予測は、例えば、逐次実行したときの逐次実行トレースの結果を参照して、予測対象のフォーク箇所候補のフォーク点からフォーク先までの実行サイクル数や該フォーク箇所候補の出現回数を参照して効果を予測してもよいし、フォーク箇所候補をそれぞれ単独で並列実行したときの並列実行性能を並列実行性能評価部２４によって評価して効果を求めてもよい。

【００６１】

続いて、フォーク箇所候補最良組合せ決定部２３は、最初に並列実行性能を評価するフォーク箇所候補の組合せ（初期組合せ）を生成し、それをフォーク箇所候補の最良組合せとする（ステップ１０２）。これは、例えば、ステップ１０１で求めたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を並列実行性能評価部２４によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとする。

【００６２】

次に、並列実行性能評価部２４は、まず、逐次処理プログラム１３を入力データ１５で逐次実行したときの逐次実行トレースを生成する（ステップ１０３）。

【００６３】

続いて、並列実行性能評価部２４は、逐次実行トレースをすべてのターム点候補を分割点として分割する（ステップ１０４）。分割された各々の部分実行トレースを、スレッド要素と呼ぶ。ターム点候補は、フォーク箇所候補が与えられれば決定することができる。例えば、特許文献３に開示されている、有効な子スレッドを生成したプロセッサが有効な子スレッドの開始アドレスの直前のアドレスの命令まで実行を完了するマルチスレッド実行方法（特許文献３の請求項１，段落００１３等参照）では、ターム点はフォーク命令のフォーク先に等しく、フォー

ク箇所候補を与えればターム点候補は一意に決定される。

【００６４】

次に、並列実行性能評価部２４は、各スレッド要素毎の実行サイクル数，スレッド要素内に出現するフォーク箇所候補とその出現位置，フォーク先のスレッド要素，スレッド要素間のメモリ依存関係等のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する（ステップ１０５）。

【００６５】

例えば、図３（ａ）を参照すると、逐次実行トレースは、フォーク箇所候補ｆ０，ｆ１，ｆ２によって、スレッド要素Ａｎ，Ｂｎ，Ｃｎに分割することができる。ここで、スレッド要素ＡｎおよびスレッドＢｎは、フォーク箇所候補ｆ０のターム点により、スレッド要素ＢｎおよびスレッドＣｎはフォーク箇所候補ｆ１のターム点により、スレッド要素ＣｎおよびスレッドＡｎはフォーク箇所候補ｆ２のターム点により決定される。

【００６６】

フォーク箇所候補ｆ０，ｆ１，ｆ２で生成可能なスレッドは、すべてスレッド要素Ａｎ，Ｂｎ，Ｃｎの組合せで構成される。例えば、図３（ｂ）に示すフォーク箇所候補ｆ０で生成されるスレッドは、スレッド要素Ｂｎ，Ｃｎ，Ａｎの順序で構成されている。同様に、図３（ｃ）～（ｆ）に示すフォーク箇所候補の他の組合せによる並列化におけるスレッドも、すべてスレッド要素Ａｎ，Ｂｎ，Ｃｎの組合せで構成される。これは、フォーク箇所候補が与えられたとき、そのフォーク箇所候補のターム点で決定されるスレッド要素の単位で、与えられたフォーク箇所候補による並列実行をシミュレートすることが可能であることを意味する。しかしながら、各スレッドの開始位置（スレッド内のフォークを実行する位置）の決定やデータ依存関係の検出のためには、逐次実行トレースを命令単位で解析する必要がある。本実施の形態では、与えられた逐次処理プログラム１３および入力データ１５に対して、各スレッド要素毎の実行サイクル数，スレッド要素内に出現するフォーク箇所候補とその出現位置，フォーク先のスレッド要素，スレッド要素間のメモリ依存度等のスレッド要素情報を一度だけ解析し、スレッド情報を各スレッド要素毎に記憶する。実際に並列実行をシミュレートする処理では

、命令単位での解析を必要としないので、並列実行性能を繰り返し評価する場合に評価速度が高速であるという特徴がある。

【００６７】

続いて、フォーク箇所候補最良組合せ決定部２３は、ステップ１０６からステップ１１３までで構成されるループＡを繰り返し実行中に、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を１つ選択する（ステップ１０６）。

【００６８】

次に、フォーク箇所候補最良組合せ決定部２３は、ステップ１０６で選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する（ステップ１０７）。

【００６９】

選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、フォーク箇所候補最良組合せ決定部２３は、ステップ１０６で選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする（ステップ１０８）。

【００７０】

一方、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補最良組合せ決定部２３は、フォーク箇所候補の最良組合せからステップ１０６で選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする（ステップ１０９）。

【００７１】

次に、フォーク箇所候補最良組合せ決定部２３は、試行組合せによる並列化の並列実行性能を、並列実行性能評価部２４によって評価する（ステップ１１０）。

【００７２】

並列実行性能評価部２４による並列実行性能の評価方法として、マルチプロセッサシステム上で実際に実行して評価する方法や、マルチプロセッサシステムのシミュレータ（図示せず）上で並列実行のシミュレーションを行って評価する方法、詳しくは、並列実行性能を評価するフォーク箇所候補の試行組合せで逐次処理

プログラム１３を並列化したとして、そのときの動作をシミュレータ上で仮想的に実行して性能を評価する方法がある。また、このとき、並列実行性能の指針としては、並列実行したときの実行サイクル数や実行時間が適当であるが、消費電力を考慮するためにプロセッサ稼働率などを評価結果算出に加味してもよい。第１の実施の形態では、並列実行性能評価部２４は、与えられたフォーク箇所候補の試行組合せに対して、ステップ１０５で記憶されたスレッド要素情報を参照して、ターム点候補で決定されるスレッド要素の単位で並列実行のシミュレーションを行って並列実行性能を評価する。

【００７３】

具体的には、並列実行性能評価部２４は、まず、並列実行性能を評価したいフォーク箇所候補の試行組合せおよび各スレッド要素毎に記憶されているフォーク箇所候補のスレッド要素情報（スレッド要素内に出現するフォーク箇所候補，フォーク先のスレッド要素）を参照して自スレッドのターム点を決定し、スレッド要素を組み合わせて自スレッドを生成する。次に、並列実行性能評価部２４は、フォーク箇所候補のスレッド要素情報（スレッド要素内に出現するフォーク箇所候補の出現位置）およびプロセッサ使用状況を参照して、そのフォーク箇所候補が生成する子スレッドの実行開始点を決定する。続いて、並列実行性能評価部２４は、各スレッド要素毎に記憶されているフォーク箇所候補のスレッド要素情報（スレッド要素間のメモリ依存関係の情報）を参照してスレッド間のメモリ依存に対する投機実行の正否を判定し、実行の取消しがあればそれをスレッドの実行開始点に反映する。

【００７４】

並列実行性能評価部２４によれば、ステップ１０４およびステップ１０５は、ある逐次実行トレースおよびターム点候補に対してあらかじめ１回実行しておけばよく、フォーク箇所候補最良組合せ決定処理のように並列実行性能を評価したいフォーク箇所候補の試行組合せを繰り返し与えられる場合には、ステップ１０６以降を実行すれば十分である。従って、例えば命令単位で並列化やメモリ依存のシミュレーションを評価毎に行う従来の並列実行シミュレーションに比べて、評価速度が高速であるという特徴がある。また、実際にマルチプロセッサ上で並列

実行を行って評価する方法にも、実働するマルチプロセッサがないと評価できない，評価したい試行組合せをマルチプロセッサで実行できる並列化プログラム１４に変換して評価結果を取り出すのにオーバヘッドが存在する等の欠点があるため、本実施形態による並列実行性能の評価方法が評価処理の高速化に有効な場合が多い。

【００７５】

続いて、フォーク箇所候補最良組合せ決定部２３は、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する（ステップ１１１）。

【００７６】

試行組合せの並列実行性能の方が良ければ、フォーク箇所候補最良組合せ決定部２３は、試行組合せを現時点でのフォーク箇所候補の最良組合せとする（ステップ１１２）。試行組合せの並列実行性能の方が悪ければ、ステップ１１２をスキップする。

【００７７】

次に、フォーク箇所候補最良組合せ決定部２３は、ループＡの繰り返し実行において、未選択のフォーク箇所候補があるかどうかを判定する（ステップ１１３）。未選択のフォーク箇所候補があれば、ステップ１０６に制御を戻し、ループＡとして繰り返し実行する。

【００７８】

未選択のフォーク箇所候補がなければ、すなわちすべてのフォーク箇所候補を選択したならば、フォーク箇所候補最良組合せ決定部２３は、直前のループＡの繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する（ステップ１１４）。

【００７９】

新たにフォーク箇所候補の最良組合せが見つかった場合には、フォーク箇所候補最良組合せ決定部２３は、ループＡの繰り返し実行に関して、すべてのフォーク箇所候補を未選択の状態にして（ステップ１１５）、ステップ１０６に制御を戻し、ループＢとして繰り返し実行する。

【００８０】

新たにフォーク箇所候補の最良組合せが見つからなかった場合には、フォーク箇所候補最良組合せ決定部２３は、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として並列化プログラム出力部２５に出力する（ステップ１１６）。

【００８１】

最後に、並列化プログラム出力部２５は、フォーク箇所候補最良組合せ決定部２３により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラム１４を生成し出力する。

【００８２】

以上説明したように、第１の実施の形態によれば、フォーク箇所候補の試行組合せに対して並列実行性能を直接評価し、それを基準にフォーク箇所候補の最良組合せを決定するので、フォーク命令を逐次処理プログラム１３により適切に挿入することが可能であり、並列実行性能のより高い並列化プログラム１４を得ることができる。

【００８３】

また、フォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序によって、その時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくため、フォーク箇所候補のすべての組合せを評価する場合に比べて、フォーク箇所候補最良組合せ決定処理の処理時間を大幅に短縮できるという効果がある。

【００８４】

[第１の実施の形態の具体例]

図４および図５を参照して、第１の実施の形態に係るプログラム並列化装置２０におけるフォーク箇所候補最良組合せ決定処理の具体例を説明する。

【００８５】

ここで、フォーク箇所候補としてｆ０，ｆ１，ｆ２，ｆ３があるとする。このときのフォーク箇所候補の組合せの総数は、まったくフォークをせずに逐次実行した場合（ϕ）を含めて１６通り存在する。

【００８６】

並列実行によって、逐次実行したとき（ϕの場合）の性能の何倍の性能が達成されるかを並列実行性能の指標として採用する。ここで、１６通りのフォーク箇所候補の組合せのそれぞれの並列実行性能を、図４に示すように仮定する。この場合、フォーク箇所候補としてｆ１およびｆ２を選択したときの並列実行性能が最も高く、逐次実行の３．８倍の性能である。

【００８７】

フォーク箇所候補の組合せは１６通りあるので、すべての組合せの並列実行性能を評価してフォーク箇所候補の最良組合せを決定する方法では、並列実行性能の評価は１６回行う必要がある。

【００８８】

次に、図４に示す具体例に対する第１の実施の形態によるフォーク箇所候補最良組合せ決定処理の動作の詳細を、図２のフローチャートも参照しつつ説明する。

【００８９】

まず、ステップ１００において、フォーク箇所候補としてｆ０，ｆ１，ｆ２，ｆ３が得られているものとする。

【００９０】

次に、ステップ１０１において、フォーク箇所候補に順序を付ける。ここで、フォーク箇所候補それぞれを単独で並列実行したときの並列実行性能を指標として、それぞれの並列実行性能が大きい順序に順序付けを行うとする。図４に示すそれぞれの並列実行性能を参照すると、ｆ０，ｆ２，ｆ１，ｆ３の順序になる。

【００９１】

続いて、ステップ１０２において、初期組合せを生成する。ステップ１０１で求めたフォーク箇所候補の順序の上位から２つを初期組合せに採用すると、｛ｆ０，ｆ２｝が初期組合せになる。｛ｆ０，ｆ２｝の並列実行性能が評価され、その結果である３．２が得られる。また、｛ｆ０，ｆ２｝は、この時点でのフォーク箇所候補の最良組合せになる。

【００９２】

次に、ステップ１０６からループＡおよびループＢが繰り返し実行される。各繰り返しにおける動作の様子を、図５に示す。

【００９３】

・ループＡ実行１回目、ループＢ実行１回目（繰り返し１－１）

まず、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１０６）。この時点では、すべてのフォーク箇所候補が未選択なので、フォーク箇所候補ｆ０が選択される。フォーク箇所候補ｆ０はフォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝に含まれているので（ステップ１０７）、フォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝からフォーク箇所候補ｆ０を取り除き、試行組合せは｛ｆ２｝となる（ステップ１０９）。次に、試行組合せの並列実行性能が評価されて、その結果である２．４が得られる（ステップ１１０）。試行組合せの並列実行性能（２．４）はフォーク箇所候補の最良組合せの並列実行性能（３．２）よりも悪いので（ステップ１１１）、フォーク箇所候補の最良組合せは｛ｆ０，ｆ２｝のままである。ループＡにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ１１３）、次にステップ１０６からループＡの２回目の実行を開始する。

【００９４】

・ループＡ実行２回目、ループＢ実行１回目（繰り返し１－２）

同様にして、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１０６）。この時点で、フォーク箇所候補ｆ０が選択済みなので、フォーク箇所候補ｆ２が選択される。フォーク箇所候補ｆ２はフォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝に含まれているので（ステップ１０７）、フォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝からフォーク箇所候補ｆ２を取り除き、試行組合せは｛ｆ０｝となる（ステップ１０９）。次に、試行組合せの並列実行性能が評価されて、その結果である２．５が得られる（ステップ１１０）。試行組合せの並列実行性能（２．５）はフォーク箇所候補の最良組合せの並列実行性能（３．２）よりも悪いので（ステップ１１１）、フォーク箇所候補の最良組合せは｛ｆ０，ｆ２｝のままである。ループＡにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ１１３）、次にステップ１０６からループＡの３回目の実行を開始する。

【００９５】

・ループＡ実行３回目、ループＢ実行１回目（繰り返し１－３）
同様にして、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１０６）。この時点で、フォーク箇所候補ｆ０およびｆ２が選択済みなので、フォーク箇所候補ｆ１が選択される。フォーク箇所候補ｆ１はフォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝に含まれていないので（ステップ１０７）、フォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝にフォーク箇所候補ｆ１を加え、試行組合せは｛ｆ０，ｆ１，ｆ２｝となる（ステップ１０８）。次に、試行組合せの並列実行性能が評価されて、その結果である３．５が得られる（ステップ１１０）。試行組合せの並列実行性能（３．５）はフォーク箇所候補の最良組合せの並列実行性能（３．２）よりも良いので（ステップ１１１）、フォーク箇所候補の最良組合せは｛ｆ０，ｆ１，ｆ２｝となる（ステップ１１２）。ループＡにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ１１３）、次にステップ１０６からループＡの４回目の実行を開始する。

【００９６】
・ループＡ実行４回目、ループＢ実行１回目（繰り返し１－４）
同様にして、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１０６）。この時点で、フォーク箇所候補ｆ０，ｆ１，ｆ２が選択済みなので、フォーク箇所候補ｆ３が選択される。フォーク箇所候補ｆ３はフォーク箇所候補の最良組合せ｛ｆ０，ｆ１，ｆ２｝に含まれていないので（ステップ１０７）、フォーク箇所候補の最良組合せ｛ｆ０，ｆ１，ｆ２｝にフォーク箇所候補ｆ３を加え、試行組合せは｛ｆ０，ｆ１，ｆ２，ｆ３｝となる（ステップ１０８）。次に、試行組合せの並列実行性能が評価されて、その結果である２．５が得られる（ステップ１１０）。試行組合せの並列実行性能（２．５）はフォーク箇所候補の最良組合せの並列実行性能（３．５）よりも悪いので（ステップ１１１）、フォーク箇所候補の最良組合せは｛ｆ０，ｆ１，ｆ２｝のままである。ループＡにおいてすべてのフォーク箇所候補を選択し（ステップ１１３）、またループＡ実行の３回目で新たにフォーク箇所候補の最良組合せが得られたので（ステップ１１４）、すべてのフォーク箇所候補

を未選択の状態にして（ステップ１１５）、次にステップ１０６からループＢ実行の２回目を開始する。

【００９７】

・ループＡ実行１回目、ループＢ実行２回目（繰り返し２－１）

まず、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１０６）。この時点では、すべてのフォーク箇所候補が未選択なので、フォーク箇所候補ｆ０が選択される。フォーク箇所候補ｆ０はフォーク箇所候補の最良組合せ｛ｆ０，ｆ１，ｆ２｝に含まれているので（ステップ１０７）、フォーク箇所候補の最良組合せ｛ｆ０，ｆ１，ｆ２｝からフォーク箇所候補ｆ０を取り除き、試行組合せは｛ｆ１，ｆ２｝となる（ステップ１０９）。次に、試行組合せの並列実行性能が評価されて、その結果である３．８が得られる（ステップ１１０）。試行組合せの並列実行性能（３．８）はフォーク箇所候補の最良組合せの並列実行性能（３．５）よりも良いので（ステップ１１１）、フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝となる（ステップ１１２）。ループＡにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ１１３）、次にステップ１０６からループＡの２回目の実行を開始する。

【００９８】

以下、詳細な説明は省略するが、図５に示すような同様な動作を行う。

【００９９】

・ループＡ実行２回目、ループＢ実行２回目（繰り返し２－２）

フォーク箇所候補ｆ２を選択して試行組合せは｛ｆ１｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。

【０１００】

・ループＡ実行３回目、ループＢ実行２回目（繰り返し２－３）

フォーク箇所候補ｆ１を選択して試行組合せは｛ｆ２｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。

【０１０１】

・ループＡ実行４回目、ループＢ実行２回目（繰り返し２－４）

フォーク箇所候補ｆ３を選択して試行組合せは｛ｆ１，ｆ２，ｆ３｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。ループＡにおいてすべてのフォーク箇所候補を選択し（ステップ１１３）、またループＡの実行の１回目で新たにフォーク箇所候補の最良組合せ｛ｆ１，ｆ２｝を得られたので（ステップ１１４）、すべてのフォーク箇所候補を未選択の状態にして（ステップ１１５）、次にステップ１０６からループＢ実行の３回目を開始する。

【０１０２】

・ループＡ実行１回目、ループＢ実行３回目（繰り返し３－１）

フォーク箇所候補ｆ０を選択して試行組合せは｛ｆ０，ｆ１，ｆ２｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。

【０１０３】

・ループＡ実行２回目、ループＢ実行３回目（繰り返し３－２）

フォーク箇所候補ｆ２を選択して試行組合せは｛ｆ１｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。

【０１０４】

・ループＡ実行３回目、ループＢ実行３回目（繰り返し３－３）

フォーク箇所候補ｆ１を選択して試行組合せは｛ｆ２｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。

【０１０５】

・ループＡ実行４回目、ループＢ実行３回目（繰り返し３－４）

フォーク箇所候補ｆ３を選択して試行組合せは｛ｆ１，ｆ２，ｆ３｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。ループＡにおいてすべてのフォーク箇所候補を選択し（ステップ１１３）、またループＡの繰り返し実行において新たにフォーク箇所候補の最良組合せが得られなかったので（ステップ１１４）、フォーク箇所候補の最良組合せである｛ｆ１，ｆ２｝をフォーク箇所候補最良組合せ決定処理の処理結果として出力し（ステップ１１６）、終了する。

【０１０６】

以上説明したように、第１の実施の形態の具体例によれば、フォーク箇所候補を

並列実行性能への効果があると予測される順序に順序付けし、その順序によって、その時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していく。フォーク箇所候補のすべての組合せについて並列実行性能を評価する方法では１６回の評価が必要なのに比べて、第１の実施の形態による並列実行性能を評価する方法では１２回の評価で済むことから明らかのように、フォーク箇所候補最良組合せ決定処理の処理時間を短縮することができるという効果がある。

【０１０７】

なお、第１の実施の形態では、マルチプロセッサシステムのシミュレータ上で並列実行のシミュレーションを行って評価するようにしたが、マルチプロセッサシステム上で実際に実行して評価するようにしてもよい。詳しくは、並列実行性能を評価するフォーク箇所候補の試行組合せに対して、実際にフォーク命令を挿入して並列化プログラムを生成し、それを実際にマルチプロセッサシステム上で実行して並列実行性能を評価する。

【０１０８】

［第２の実施の形態］

マルチスレッド実行方法としてフォーク１回モデルを採用した場合、スレッドはその生存中に高々１回に限って有効な子スレッドを生成することを保証する必要がある。これには、非特許文献１等で開示されているように、逐次処理プログラムから並列化プログラムを生成するコンパイルの段階で静的に制限する方法や、特許文献３で開示されているように、親スレッド実行中に選択することにより動的に保証する方法があるが、いずれの方式においても、フォーク１回モデル上で、同時に実行することのできないフォーク箇所候補の組合せが存在する。すなわち、フォーク１回モデルを仮定した場合、互いに同時に実行可能なフォーク箇所候補の試行組合せのみに対して並列実行性能の評価を行うことにすると、フォーク箇所候補最良組合せ決定処理の処理時間の短縮を期待することができる。

【０１０９】

図６を参照すると、本発明の第２の実施の形態に係るプログラム並列化装置２０’は、入力される逐次処理プログラム１３の制御フローおよびデータフローの解

析を行う制御・データフロー解析部２１と、制御・データフロー解析部２１による制御フローおよびデータフローの解析結果を参照して逐次処理プログラム１３のフォーク箇所候補を決定するフォーク箇所候補決定部２２と、フォーク箇所候補決定部２２により決定されたフォーク箇所候補からフォーク１回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを並列実行性能評価部２４に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部２３’と、フォーク箇所候補最良組合せ決定部２３’より与えられるフォーク箇所候補の試行組合せにより逐次処理プログラム１３を並列化したときの並列実行性能を、ある入力データ１５に対して評価する並列実行性能評価部２４と、フォーク箇所候補最良組合せ決定部２３’により決定されたフォーク箇所候補の最良組合せに基づいて並列化プログラム１４を生成し出力する並列化プログラム出力部２５とを備える。

【０１１０】

図７を参照すると、フォーク箇所候補決定部２２，フォーク箇所候補最良組合せ決定部２３’，および並列実行性能評価部２４によるフォーク箇所候補最良組合せ決定処理は、フォーク箇所候補生成ステップ１２０と、フォーク箇所候補順序付けステップ１２１と、フォーク箇所候補初期組合せ生成ステップ１２２と、逐次実行トレース生成ステップ１２３と、スレッド要素分割ステップ１２４と、スレッド要素情報解析・記憶ステップ１２５と、フォーク箇所候補選択ステップ１２６と、フォーク箇所候補最良組合せ判定ステップ１２７と、フォーク箇所候補加入・試行組合せ設定ステップ１２８と、キャンセルフォーク箇所候補削除ステップ１３７と、フォーク箇所候補削除・試行組合せ設定ステップ１２９と、並列実行性能評価ステップ１３０と、並列実行性能比較ステップ１３１と、キャンセルフォーク箇所候補除去ステップ１３８と、最良組合せ設定ステップ１３２と、全フォーク箇所候補選択判定ステップ１３３と、新最良組合せ発見判定ステップ１３４と、全フォーク箇所候補未選択設定ステップ１３５と、フォーク箇所候補最良組合せ出力ステップ１３６とからなる。

【０１１１】

図７のフォーク箇所候補最良組合せ決定処理は、フォーク１回制限の保証方法として、特許文献３に開示されている、親スレッドのフォーク命令毎に当該親スレッドから生成された子スレッドが既に存在する場合には、その子スレッドをキャンセルすることで、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する１つのフォーク命令を親スレッド実行中に選択する方法（特許文献３の請求項２，段落００１４等参照）を採用すると仮定した場合の動作である。

【０１１２】

フォーク１回モデルの実現方法には、既述したように、主に逐次処理プログラムから並列化プログラムを生成するコンパイル段階で静的に制限する方法と、親スレッド実行中に選択することにより動的に保証する方法がある。後者の具体的な方法には、親スレッドのフォーク命令毎に当該親スレッドから生成された子スレッドがすでに存在する場合には、その子スレッドをキャンセルすることで、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する１つのフォーク命令を親スレッド実行中に選択する方法があり、第２の実施の形態では、この方法を仮定する。

【０１１３】

この方法によると、フォーク箇所候補ｆ０とフォーク箇所候補ｆ１とを同時に選択した場合、フォーク箇所候補ｆ１は必ずフォーク箇所候補ｆ０によりキャンセルされると仮定すると、第２の実施の形態のフローチャートのループＡのある繰り返しにおいて、フォーク箇所候補ｆ１を選択し、それをフォーク箇所候補ｆ０を含む試行組合せに加える場合を考える。この場合、効果を評価したいフォーク箇所候補ｆ１は並列実行時にフォーク箇所候補ｆ０によりキャンセルされてしまう。ステップ１３７は、キャンセルの原因であるフォーク箇所候補ｆ０を試行組合せからあらかじめ取り除いて、フォーク箇所候補ｆ１が実行されるようにする処理に相当する。

【０１１４】

逆に、フローチャートのループＡのある繰り返しにおいてフォーク箇所候補ｆ０を選択し、それをフォーク箇所候補ｆ１を含む試行組合せに加える場合を考える

。この場合、効果を評価したいフォーク箇所候補ｆ０はキャンセルされないが、フォーク箇所候補ｆ１は必ずキャンセルされるため、フォーク箇所候補ｆ１が試行組合せに入っている意味がない。ステップ１３８は、この必ずキャンセルされて意味をなさないフォーク箇所候補をあらかじめ取り除く処理に相当する。

【０１１５】

次に、このように構成された第２の実施の形態に係るプログラム並列化装置２０’の動作について説明する。

【０１１６】

まず、制御・データフロー解析部２１は、逐次処理プログラム１３の制御フローおよびデータフローを解析する。

【０１１７】

次に、フォーク箇所候補決定部２２は、制御・データフロー解析部２１による制御フローおよびデータフローの解析結果を参照して、フォーク箇所候補を生成する（ステップ１２０）。

【０１１８】

次に、フォーク箇所候補最良組合せ決定部２３’は、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける（ステップ１２１）。効果の予測は、例えば、逐次実行したときの逐次実行トレースの結果を参照して、予測対象のフォーク箇所候補のフォーク元からフォーク先までの実行サイクル数および該フォーク箇所候補の出現回数を参照して効果を予測してもよいし、フォーク箇所候補をそれぞれ単独で並列実行したときの並列実行性能を並列実行性能評価部２４によって評価して効果を求めてもよい。さらに、フォーク１回制限を保証するためのフォークのキャンセルを発生させるフォーク箇所候補間の関係を求める。これは、逐次実行トレースの結果や制御フローグラフを参照して求めることができる。例えば、逐次処理プログラム１３の実行フロー上で、あるフォーク箇所候補ｆａのフォーク点からフォーク先までの間に、あるフォーク箇所候補ｆｂのフォーク点がある場合、実行時にフォーク箇所候補ｆａはフォーク箇所候補ｆｂによってキャンセルされる。

【０１１９】

続いて、フォーク箇所候補最良組合せ決定部２３’は、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする（ステップ１２２）。これは、例えば、ステップ１２１で求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を並列実行性能評価部２４によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとする。

【０１２０】

並列実行性能評価部２４は、まず、逐次処理プログラム１３を入力データ１５で逐次実行したときの逐次実行トレースを生成する（ステップ１２３）。

【０１２１】

次に、並列実行性能評価部２４は、逐次実行トレースをすべてのターム点候補を分割点として分割する（ステップ１２４）。

【０１２２】

続いて、並列実行性能評価部２４は、各スレッド要素毎の実行サイクル数，スレッド要素内に出現するフォーク箇所候補とその出現位置，フォーク先のスレッド要素，スレッド要素間のメモリ依存関係等のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する（ステップ１２５）。

【０１２３】

次に、フォーク箇所候補最良組合せ決定部２３’は、ステップ１２６からステップ１３３までで構成されるループＡを繰り返し実行中に、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を１つ選択する（ステップ１２６）。

【０１２４】

続いて、フォーク箇所候補最良組合せ決定部２３’は、ステップ１２６で選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する（ステップ１２７）。

【０１２５】

選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、フォーク箇所候補最良組合せ決定部２３’は、フォーク箇所候補の最良組

合せにステップ１２６で選択されたフォーク箇所候補を加え、そのフォーク箇所候補の組合せを試行組合せとする（ステップ１２８）。

【０１２６】

次に、フォーク箇所候補最良組合せ決定部２３’は、ステップ１２６で選択されたフォーク箇所候補をキャンセルするフォーク箇所候補を試行組合せから取り除く（ステップ１３７）。

【０１２７】

一方、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補最良組合せ決定部２３’は、フォーク箇所候補の最良組合せからステップ１２６で選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする（ステップ１２９）。

【０１２８】

次に、フォーク箇所候補最良組合せ決定部２３’は、試行組合せによる並列化の並列実行性能を、並列実行性能評価部２４によって評価する（ステップ１３０）。

【０１２９】

続いて、フォーク箇所候補最良組合せ決定部２３’は、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する（ステップ１３１）。試行組合せの並列実行性能の方が悪ければ、ステップ１３８およびステップ１３２をスキップする。

【０１３０】

試行組合せの並列実行性能の方が良ければ、フォーク箇所候補最良組合せ決定部２３’は、ステップ１２６で選択されたフォーク箇所候補がキャンセルするフォーク箇所候補を試行組合せから取り除く（ステップ１３８）。

【０１３１】

次に、フォーク箇所候補最良組合せ決定部２３’は、試行組合せを現時点でのフォーク箇所候補の最良組合せとする（ステップ１３２）。

【０１３２】

次に、フォーク箇所候補最良組合せ決定部２３’は、ループＡの繰り返し実行に

おいて、未選択のフォーク箇所候補があるかどうかを判定する（ステップ１３３）。未選択のフォーク箇所候補があれば、ステップ１２６に制御を戻し、ループＡとして繰り返し実行する。

【０１３３】

未選択のフォーク箇所候補がなければ、すなわちすべてのフォーク箇所候補を選択したならば、フォーク箇所候補最良組合せ決定部２３’は、直前のループＡの繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する（ステップ１３４）。

【０１３４】

新たにフォーク箇所候補の最良組合せが見つかった場合には、フォーク箇所候補最良組合せ決定部２３’は、ループＡの繰り返し実行に関して、すべてのフォーク箇所候補を未選択の状態にして（ステップ１３５）、ステップ１２６に制御を戻し、ループＢとして繰り返し実行する。

【０１３５】

新たにフォーク箇所候補の最良組合せが見つからなかった場合には、フォーク箇所候補最良組合せ決定部２３’は、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として並列化プログラム出力部２５に出力する（ステップ１３６）。

【０１３６】

最後に、並列化プログラム出力部２５は、フォーク箇所候補最良組合せ決定部２３’により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラム１４を生成し出力する。

【０１３７】

以上説明したように、第２の実施の形態によれば、マルチスレッド実行方法としてフォーク１回モデルを採用した場合に、互いに同時に実行可能なフォーク箇所候補の試行組合せのみを対象に並列実行性能を評価するので、フォーク箇所候補最良組合せ決定処理の処理時間を、第１の実施の形態に比べてさらに短縮することができるという効果が得られる。

【０１３８】

［第２の実施の形態の具体例］

第２の実施の形態によるフォーク箇所候補最良組合せ決定処理の具体例を、図４および図８を参照して説明する。

【０１３９】

ここで、フォーク箇所候補としてｆ０，ｆ１，ｆ２，ｆ３があるとする。このときのフォーク箇所候補の組合せの総数は、まったくフォークをせずに逐次実行した場合（ϕ）を含めて１６通り存在する。

【０１４０】

並列実行によって、逐次実行したとき（ϕの場合）の性能の何倍の性能が達成されるかを並列実行性能の指標として採用する。ここで、１６通りのフォーク箇所候補の組合せのそれぞれの並列実行性能を、図４に示すように仮定する。この場合、フォーク箇所候補としてｆ１およびｆ２を選択したときの並列実行性能が最も高く、逐次実行の３．８倍の性能である。

【０１４１】

次に、図４に示す具体例に対する第２の実施の形態によるフォーク箇所候補最良組合せ決定処理の動作の詳細を、図７のフローチャートも参照しつつ説明する。

【０１４２】

まず、ステップ１２０において、フォーク箇所候補としてｆ０，ｆ１，ｆ２，ｆ３が得られているとする。

【０１４３】

次に、ステップ１２１において、フォーク箇所候補に順序を付ける。ここで、フォーク箇所候補それぞれを単独で並列実行したときの並列実行性能を指標として、それぞれの並列実行性能が大きい順序に順序付けを行うとする。図４に示すそれぞれの並列実行性能を参照すると、ｆ０，ｆ２，ｆ１，ｆ３の順序になる。次に、フォーク箇所候補間のキャンセル関係を求める。ここでは、フォーク箇所候補ｆ０がフォーク箇所候補ｆ１をキャンセルし、フォーク箇所候補ｆ２がフォーク箇所候補ｆ３をキャンセルする関係が存在すると仮定する。

【０１４４】

続いて、ステップ１２２において、初期組合せを生成する。ステップ１２１で求

めたフォーク箇所候補の順序の上位から２つを初期組合せに採用すると、｛ｆ０，ｆ２｝が初期組合せになる。｛ｆ０，ｆ２｝の並列実行性能が評価され、その結果である３．２が得られる。また、｛ｆ０，ｆ２｝は、この時点でのフォーク箇所候補の最良組合せになる。

【０１４５】

次に、ステップ１２６からループＡおよびループＢを繰り返し実行される。各繰り返しにおける動作の様子を、図８に示す。

【０１４６】

・ループＡ実行１回目、ループＢ実行１回目（繰り返し１－１）

まず、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１２６）。この時点では、すべてのフォーク箇所候補が未選択なので、フォーク箇所候補ｆ０が選択される。フォーク箇所候補ｆ０はフォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝に含まれているので（ステップ１２７）、フォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝からフォーク箇所候補ｆ０を取り除き、試行組合せは｛ｆ２｝となる（ステップ１２９）。次に、試行組合せの並列実行性能が評価されて、その結果である２．４が得られる（ステップ１３０）。試行組合せの並列実行性能（２．４）はフォーク箇所候補の最良組合せの並列実行性能（３．２）よりも悪いので（ステップ１３１）、フォーク箇所候補の最良組合せは｛ｆ０，ｆ２｝のままである。ループＡにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ１３３）、次にステップ１２６からループＡの２回目の実行を開始する。

【０１４７】

・ループＡ実行２回目、ループＢ実行１回目（繰り返し１－２）

同様にして、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１２６）。この時点で、フォーク箇所候補ｆ０が選択済みなので、フォーク箇所候補ｆ２が選択される。フォーク箇所候補ｆ２はフォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝に含まれているので（ステップ１２７）、フォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝からフォーク箇所候補ｆ２を取り除き、試行組合せは｛ｆ０｝となる（ステップ１２９）。

次に、試行組合せの並列実行性能が評価されて、その結果である２．５が得られる（ステップ１３０）。試行組合せの並列実行性能（２．５）はフォーク箇所候補の最良組合せの並列実行性能（３．２）よりも悪いので（ステップ１３１）、フォーク箇所候補の最良組合せは｛ｆ０，ｆ２｝のままである。ループＡにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ１３３）、次にステップ１２６からループＡの３回目の実行を開始する。

【０１４８】

・ループＡ実行３回目、ループＢ実行１回目（繰り返し１－３）

同様にして、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１２６）。この時点で、フォーク箇所候補ｆ０およびｆ２が選択済みなので、フォーク箇所候補ｆ１が選択される。フォーク箇所候補ｆ１はフォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝に含まれていないので（ステップ１２７）、フォーク箇所候補の最良組合せ｛ｆ０，ｆ２｝にフォーク箇所候補ｆ１を加え、試行組合せは｛ｆ０，ｆ１，ｆ２｝となる（ステップ１２８）。次に、選択されたフォーク箇所候補ｆ１がフォーク箇所候補ｆ０にキャンセルされるので、試行組合せ｛ｆ０，ｆ１，ｆ２｝からフォーク箇所候補ｆ０は取り除かれて、試行組合せが｛ｆ１，ｆ２｝となる（ステップ１３７）。次に、試行組合せの並列実行性能が評価されて、その結果である３．８が得られる（ステップ１３０）。試行組合せの並列実行性能（３．８）はフォーク箇所候補の最良組合せの並列実行性能（３．２）よりも良いので（ステップ１３１）、フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝となる（ステップ１３２）。ループＡにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ１３３）、次にステップ１２６からループＡの４回目の実行を開始する。

【０１４９】

・ループＡ実行４回目、ループＢ実行１回目（繰り返し１－４）

同様にして、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１２６）。この時点で、フォーク箇所候補ｆ０，ｆ１，ｆ２が選択済みなので、フォーク箇所候補ｆ３が選択される。フォーク箇所候補ｆ３はフォーク箇所候補の最良組合せ｛ｆ１，ｆ２｝に含ま

れていないので（ステップ１２７）、フォーク箇所候補の最良組合せ｛ｆ１，ｆ２｝にフォーク箇所候補ｆ３を加え、試行組合せは｛ｆ１，ｆ２，ｆ３｝となる（ステップ１２８）。次に、選択されたフォーク箇所候補ｆ３がフォーク箇所候補ｆ２にキャンセルされるので、試行組合せ｛ｆ１，ｆ２，ｆ３｝からフォーク箇所候補ｆ２は取り除かれて、試行組合せが｛ｆ１，ｆ３｝となる（ステップ１３７）。次に、試行組合せの並列実行性能が評価されて、その結果である２．８が得られる（ステップ１３０）。試行組合せの並列実行性能（２．８）はフォーク箇所候補の最良組合せ｛ｆ１，ｆ２｝の並列実行性能（３．８）よりも悪いので（ステップ１３１）、フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。ループＡにおいてすべてのフォーク箇所候補を選択し（ステップ１３３）、またループＡの実行の３回目で新たにフォーク箇所候補の最良組合せを得られたので（ステップ１３４）、すべてのフォーク箇所候補を未選択の状態にして（ステップ１３５）、次にステップ１２６からループＢ実行の２回目を開始する。

【０１５０】

・ループＡ実行１回目、ループＢ実行２回目（繰り返し２－１）

まず、ループＡの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ１２６）。この時点ではすべてのフォーク箇所候補が未選択なので、フォーク箇所候補ｆ０が選択される。フォーク箇所候補ｆ０はフォーク箇所候補の最良組合せ｛ｆ１，ｆ２｝に含まれていないので（ステップ１２７）、フォーク箇所候補の最良組合せ｛ｆ１，ｆ２｝にフォーク箇所候補ｆ０を加え、試行組合せは｛ｆ０，ｆ１，ｆ２｝となる（ステップ１２８）。次に、試行組合せの並列実行性能が評価されて、その結果である３．５が得られる（ステップ１３０）。試行組合せの並列実行性能（３．５）はフォーク箇所候補の最良組合せ｛ｆ１，ｆ２｝の並列実行性能（３．８）よりも悪いので（ステップ１３１）、フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。ループＡにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ１３３）、次にステップ１２６からループＡの２回目の実行を開始する。

【０１５１】

以下、詳細な説明は省略するが、図８に示すような同様な動作を行う。

【０１５２】

・ループＡ実行２回目、ループＢ実行２回目（繰り返し２－２）

フォーク箇所候補ｆ２を選択して試行組合せは｛ｆ１｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。

【０１５３】

・ループＡ実行３回目、ループＢ実行２回目（繰り返し２－３）

フォーク箇所候補ｆ１を選択して試行組合せは｛ｆ２｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。

【０１５４】

・ループＡ実行４回目、ループＢ実行２回目（繰り返し２－４）

フォーク箇所候補ｆ３を選択して試行組合せは｛ｆ１，ｆ３｝となる。フォーク箇所候補の最良組合せは｛ｆ１，ｆ２｝のままである。ループＡにおいてすべてのフォーク箇所候補を選択し（ステップ１３３）、またループＡの繰り返し実行において新たにフォーク箇所候補の最良組合せが得られなかったので（ステップ１３４）、フォーク箇所候補の最良組合せである｛ｆ１，ｆ２｝をフォーク箇所候補最良組合せ決定処理の処理結果として出力し（ステップ１３６）、終了する。

【０１５５】

以上説明したように第２の実施の形態の具体例によれば、マルチスレッド実行方法としてフォーク１回モデルを採用した場合に、互いに同時に実行可能なフォーク箇所候補の試行組合せのみを対象に並列実行性能を評価するので、８回の評価でフォーク箇所候補最良組合せ決定処理が終了することから明らかのように、第１の実施の形態の具体例に比べて、さらにフォーク箇所候補最良組合せ決定処理の処理時間を短縮できるという効果が得られる。

【０１５６】

［第３の実施の形態］

フォーク箇所候補の組合せの総数はフォーク箇所候補の数に対して指数関数的に増加するため、通常、フォーク箇所候補の数が増加するにつれてフォーク箇所候

補最良組合せ決定処理の処理時間も急速に増加してしまう。これに対し、フォーク箇所候補を適当数にグループ分割し、各グループ毎に独立にフォーク箇所候補最良組合せ決定処理を行うと、フォーク箇所候補の組合せの総数を削減することができるため、フォーク箇所候補の数が多い場合でもフォーク箇所候補最良組合せ決定処理を高速に行うことができる。このとき、並列実行性能のより高いフォーク箇所候補を得るためにはフォーク箇所候補のグループ分割が特に重要であり、独立なフォーク箇所候補最良組合せ決定処理を可能にするために、グループ間ではなるべく互いに影響を及ぼし合わないようにグループ分割することが必要である。第３の実施の形態では、フォーク箇所候補間のキャンセル関係を参照し、キャンセル関係があれば関係が強く、なければ弱いとしてグループ分割することでこれを達成する。

【０１５７】

図９を参照すると、本発明の第３の実施の形態に係るプログラム並列化装置２０”は、入力される逐次処理プログラム１３の制御フローおよびデータフローの解析を行う制御・データフロー解析部２１と、制御・データフロー解析部２１による制御フローおよびデータフローの解析結果を参照して逐次処理プログラム１３のフォーク箇所候補を決定するフォーク箇所候補決定部２２と、フォーク箇所候補決定部２２により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようにフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を並列実行性能評価部２４によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定するフォーク箇所候補最良組合せ決定部２３”と、フォーク箇所候補最良組合せ決定部２３”より与えられるフォーク箇所候補の試行組合せにより逐次処理プログラム１３を並列化したときの並列実行性能を、ある入力データ１５に対して評価する並列実行性能評価部２４と、フォーク箇所候補最良組合せ決定部２３”により決定されたフォーク箇所候補の試行組合せに基づいて並列

化プログラム１４を生成し出力する並列化プログラム出力部２５とを備える。

【０１５８】

図１０（ａ）を参照すると、フォーク箇所候補最良組合せ決定部２３”によるフォーク箇所候補最良組合せ決定処理は、グループ分割処理呼び出しステップ１４０と、グループに対するフォーク箇所候補最良組合せ決定処理ステップ１４１と、全グループ処理判定ステップ１４２と、結果出力ステップ１４３とからなる。

【０１５９】

図１０（ｂ）を参照すると、フォーク箇所候補最良組合せ決定部２３”によるフォーク箇所候補のグループ分割処理は、フォーク箇所候補数・分割下限数比較ステップ１４６と、フォーク箇所候補分割ステップ１４７と、集合のグループへの分割ステップ１４８と、グループ分割処理呼び出しステップ１４９と、グループ分割処理呼び出しステップ１５０とからなる。

【０１６０】

次に、このように構成された第３の実施の形態に係るプログラム並列化装置２０”の動作について説明する。

【０１６１】

制御・データフロー解析部２１およびフォーク箇所候補決定部２２の処理は、第１および第２の実施の形態における動作と全く同様であるので、説明を省略する。

【０１６２】

フォーク箇所候補決定部２２の処理が終了すると、フォーク箇所候補最良組合せ決定部２３”は、フォーク箇所候補決定部２２により決定された全フォーク箇所候補の集合をＦとして、図４（ｂ）のフローチャートに示すフォーク箇所候補のグループ分割処理を呼び出す（ステップ１４０）。

【０１６３】

フォーク箇所候補のグループ分割処理では、フォーク箇所候補最良組合せ決定部２３”は、フォーク箇所候補間のキャンセル関係を参照して２分木状にフォーク箇所候補をグループに分割するものとする。なお、分割下限数Ｍおよび指定数Ｎは適切に与えられる数であり、例えばＭ＝３００、Ｎ＝Ｍ／２とすることができ

る。

【０１６４】

ステップ１４０から同図（ｂ）に示すフォーク箇所候補のグループ分割処理が呼び出されると、フォーク箇所候補最良組合せ決定部２３”は、まず、与えられたフォーク箇所候補の集合Ｆに属するフォーク箇所候補の数が指定された分割下限数Ｍより大きければ（ステップ１４６）、ステップ１４７から集合Ｆのグループ分割処理を開始する。小さければ、フォーク箇所候補最良組合せ決定部２３”は、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元（ステップ１４０，ステップ１４９またはステップ１５０）にリターンする。

【０１６５】

次に、フォーク箇所候補最良組合せ決定部２３”は、自身をキャンセルするフォーク箇所候補の数が指定数Ｎより大きいフォーク箇所候補の集合を集合Ｆから分割して新たにグループを生成する（ステップ１４７）。

【０１６６】

続いて、フォーク箇所候補最良組合せ決定部２３”は、集合Ｆをさらに２つのグループＦａおよびＦｂにグループ分割する（ステップ１４８）。このとき、グループＦａに属するフォーク箇所候補をキャンセルするフォーク箇所候補の中でグループＦｂに属するフォーク箇所候補の数、およびグループＦｂに属するフォーク箇所候補をキャンセルするフォーク箇所候補の中でグループＦａに属するフォーク箇所候補の数の総和をなるべく少なくするようにグループ分割を行う。これは、互いになるべくフォーク箇所候補間で相関がないようにグループ分割するためである。このとき、２つのグループＦａおよびＦｂにそれぞれ属するフォーク箇所候補の数は同等の数である必要はなく、上記総和が小さくなる限り差があってもよい。

【０１６７】

次に、フォーク箇所候補最良組合せ決定部２３”は、グループＦａを集合Ｆとしてフォーク箇所候補のグループ分割処理を再帰的に呼び出してグループＦａのグループ分割を行う（ステップ１４９）。

【０１６８】

同様に、フォーク箇所候補最良組合せ決定部２３”は、グループＦｂを集合Ｆとしてフォーク箇所候補のグループ分割処理を再帰的に呼び出してグループＦｂのグループ分割を行う（ステップ１５０）。

【０１６９】

この後、フォーク箇所候補最良組合せ決定部２３”は、フォーク箇所候補のグループ分割処理を呼び出した元（ステップ１４０，ステップ１４９またはステップ１５０）にリターンする。

【０１７０】

次に、フォーク箇所候補最良組合せ決定部２３”は、ステップ１４０で分割されたフォーク箇所候補のグループの中で、フォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行う（ステップ１４１）。このフォーク箇所候補最良組合せ決定処理は、第１の実施の形態あるいは第２の実施の形態におけるフォーク箇所候補最良組合せ決定処理を適用してもよいし、各グループ毎のフォーク箇所候補の数が十分少ない場合は、グループ内のフォーク箇所候補のすべての組合せについて並列実行性能の評価を行ってフォーク箇所候補の最良組合せを決定してもよい。第１の実施の形態または第２の実施の形態におけるフォーク箇所候補最良組合せ決定処理を適用する場合、フォーク箇所候補の初期組合せをグループ内のフォーク箇所候補のみから構成することもできる。

【０１７１】

続いて、フォーク箇所候補最良組合せ決定部２３”は、全グループの処理が完了したかどうかを判定し（ステップ１４２）、未処理のグループがあるならば、ステップ１４１に制御を戻して、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返す。

【０１７２】

全グループの処理が完了した場合は、フォーク箇所候補最良組合せ決定部２３”は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力する（ステップ１４３）。

【０１７３】

［第３の実施の形態の具体例］

図１１を参照して、第３の実施の形態に係るプログラム並列化装置２０”におけるフォーク箇所候補のグループ分割処理の具体例を説明する。

【０１７４】

全フォーク箇所候補の集合を集合Ｆとし、集合Ｆのグループ分割を行うとする。集合Ｆのフォーク箇所候補の数が分割下限数Ｍ以上とすると（ステップ１４６）、まず集合Ｆから自身をキャンセルするフォーク箇所候補の数が指定数Ｎ以上のフォーク箇所候補を分離して、新たなグループＦ０とする（ステップ１４７）。次に、集合Ｆを２つのグループＦ０－ａとＦ０－ｂとに分割する（ステップ１４８）。すなわち、集合Ｆは、３つのフォーク箇所候補のグループＦ０，Ｆ０－ａ，Ｆ０－ｂに分割される。次に、グループＦ０－ａおよびグループＦ０－ｂに対して再びフォーク箇所候補のグループ分割処理を行う（ステップ１４９および１５０）。

【０１７５】

同様に、グループＦ０－ａのフォーク箇所候補の数が分割下限数Ｍ以上とすると、グループＦ０－ａは、グループＦ１，Ｆ１－ａ，Ｆ１－ｂに分割される。グループＦ１－ａおよびＦ１－ｂに対して再びフォーク箇所候補のグループ分割処理を行う。

【０１７６】

同様に、グループＦ１－ａのフォーク箇所候補の数が分割下限数Ｍ以上とすると、グループＦ１－ａは、グループＦ２，Ｆ２－ａ，Ｆ２－ｂに分割される。グループＦ２－ａおよびＦ２－ｂに対して再びフォーク箇所候補のグループ分割処理を行う。

【０１７７】

次に、グループＦ２－ａのフォーク箇所候補の数が分割下限数Ｍ以下とすると、これ以上分割されずグループＦ２－ａをグループＦ３とする。

【０１７８】

次に、グループＦ２－ｂのフォーク箇所候補の数が分割下限数Ｍ以下とすると、

これ以上分割されずグループＦ２－ｂをグループＦ４とする。

【０１７９】

同様に、グループＦ１－ｂのフォーク箇所候補の数が分割下限数Ｍ以下とすると、これ以上分割されずグループＦ１－ｂをグループＦ５とする。

【０１８０】

次に、グループＦ０－ｂのフォーク箇所候補の数が分割下限数Ｍ以上とすると、グループＦ０－ｂは、グループＦ６，Ｆ６－ａ，Ｆ６－ｂに分割される。グループＦ６－ａおよびグループＦ６－ｂに対して再びフォーク箇所候補のグループ分割処理を行う。

【０１８１】

次に、グループＦ６－ａのフォーク箇所候補の数が分割下限数Ｍ以下とすると、これ以上分割されずグループＦ６－ａをグループＦ７とする。

【０１８２】

次に、グループＦ６－ｂのフォーク箇所候補の数が分割下限数Ｍ以下とすると、これ以上分割されずグループＦ６－ｂをグループＦ８とする。

【０１８３】

以上により、全フォーク箇所候補の集合Ｆａが、相互に及ぼす影響の少ないフォーク箇所候補のグループＦ０～Ｆ８に分割された。

【０１８４】

以上に説明したように、第３の実施の形態によれば、フォーク箇所候補をグループ間では互いになるべく影響を及ぼし合わないように適当数のグループに分割して、各グループ毎に独立にフォーク箇所候補最良組合せ決定処理を行うので、フォーク箇所候補の組合せの総数を削減することができ、フォーク箇所候補の数が多い場合でもフォーク箇所候補最良組合せ決定処理を高速に行うことができる。

【０１８５】

［第４の実施の形態］

図１２は、本発明の第４の実施の形態に係るプログラム並列化装置２０の構成を示すブロック図である。本実施の形態に係るプログラム並列化装置２０は、図１に示した第１の実施の形態に係るプログラム並列化装置２０に対してプログラム

並列化プログラム４００を付加するようにした点だけが異なる。したがって、その他の特に言及しない部分には同一符号を付して、それらの詳しい説明を省略する。

【０１８６】

プログラム並列化プログラム４００は、コンピュータでなるプログラム並列化装置２０に読み込まれ、プログラム並列化装置２０の動作を、制御・データフロー解析部２１，フォーク箇所候補決定部２２，フォーク箇所候補最良組合せ決定部２３，並列実行性能評価部２４，および並列化プログラム出力部２５として制御する。プログラム並列化プログラム４００の制御によるプログラム並列化装置２０の動作は、第１の実施の形態におけるプログラム並列化装置２０の動作と全く同様になるので、その詳しい説明を割愛する。

【０１８７】

［第５の実施の形態］

図１３は、本発明の第５の実施の形態に係るプログラム並列化装置２０’の構成を示すブロック図である。本実施の形態に係るプログラム並列化装置２０’は、図６に示した第２の実施の形態に係るプログラム並列化装置２０’に対してプログラム並列化プログラム４００’を付加するようにした点だけが異なる。したがって、その他の特に言及しない部分には同一符号を付して、それらの詳しい説明を省略する。

【０１８８】

プログラム並列化プログラム４００’は、コンピュータでなるプログラム並列化装置２０’に読み込まれ、プログラム並列化装置２０’の動作を、制御・データフロー解析部２１，フォーク箇所候補決定部２２，フォーク箇所候補最良組合せ決定部２３’，並列実行性能評価部２４，および並列化プログラム出力部２５として制御する。プログラム並列化プログラム４００’の制御によるプログラム並列化装置２０’の動作は、第２の実施の形態におけるプログラム並列化装置２０’の動作と全く同様になるので、その詳しい説明を割愛する。

【０１８９】

［第６の実施の形態］

図１４は、本発明の第６の実施の形態に係るプログラム並列化装置２０”の構成を示すブロック図である。本実施の形態に係るプログラム並列化装置２０”は、図９に示した第３の実施の形態に係るプログラム並列化装置２０”に対してプログラム並列化プログラム４００”を付加するようにした点だけが異なる。したがって、その他の特に言及しない部分には同一符号を付して、それらの詳しい説明を省略する。

【０１９０】

プログラム並列化プログラム４００”は、コンピュータでなるプログラム並列化装置２０”に読み込まれ、プログラム並列化装置２０”の動作を、制御・データフロー解析部２１，フォーク箇所候補決定部２２，フォーク箇所候補最良組合せ決定部２３”，並列実行性能評価部２４，および並列化プログラム出力部２５として制御する。プログラム並列化プログラム４００”の制御によるプログラム並列化装置２０”の動作は、第３の実施の形態におけるプログラム並列化装置２０”の動作と全く同様になるので、その詳しい説明を割愛する。

【０１９１】

【発明の効果】

第１の効果は、フォーク命令を逐次処理プログラムにより適切に挿入することができ、並列実行性能のより高い並列化プログラムを得ることができることである。その理由は、フォーク箇所候補の試行組合せに対して並列実行性能を直接評価した結果を基準にフォーク箇所候補の最良組合せを決定するからである。

【０１９２】

第２の効果は、フォーク箇所候補のすべての組合せに対して並列実行性能を評価する場合に比べて、フォーク箇所候補最良組合せ決定処理の処理時間を大幅に短縮できるということである。その理由は、フォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくためである。

【０１９３】

第３の効果は、フォーク箇所候補最良組合せ決定処理の処理時間をさらに短縮で

きるということである。その理由は、マルチスレッド実行方法としてフォーク１回モデルを採用した場合に、互いに同時に実行可能なフォーク箇所候補の試行組合せのみを対象に並列実行性能を評価するからである。

【０１９４】

第４の効果は、フォーク箇所候補の組合せの総数を削減することができ、フォーク箇所候補の数が多い場合でもフォーク箇所候補最良組合せ決定処理を高速に行うことができることである。その理由は、フォーク箇所候補をグループ間では互いになるべく影響を及ぼし合わないように適当数のグループに分割して各グループ毎に独立にフォーク箇所候補最良組合せ決定処理を行うからである。

【０１９５】

第５の効果は、並列実行性能を高速に評価することが可能であり、従ってフォーク箇所候補最良組合せ決定処理を高速に行うことができることである。その理由は、ターム点候補で決定されるスレッド要素の単位で並列実行のシミュレーションを行うようにしたからである。

【図面の簡単な説明】

【図１】

本発明の第１の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

【図２】

本発明の第１の実施の形態におけるフォーク箇所候補最良組合せ決定処理を示すフローチャートである。

【図３】

フォーク１回モデルにおけるマルチスレッド実行方法の一例を示す図である。

【図４】

第１の実施の形態によるフォーク箇所候補最良組合せ決定処理の具体例を説明する図である。

【図５】

第１の本実施の形態によるフォーク箇所候補最良組合せ決定処理の具体例を説明する図である。

【図６】

本発明の第２の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

【図７】

本発明の第２の実施の形態におけるフォーク箇所候補最良組合せ決定処理を示すフローチャートである。

【図８】

第２の実施の形態によるフォーク箇所候補最良組合せ決定処理の具体例を説明する図である。

【図９】

本発明の第３の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

【図１０】

本発明の第３の実施の形態におけるフォーク箇所候補最良組合せ決定処理を示すフローチャートである。

【図１１】

本発明の第３の実施の形態におけるフォーク箇所候補のグループ分割の一例を示す図である。

【図１２】

本発明の第４の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

【図１３】

本発明の第５の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

【図１４】

本発明の第６の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

【図１５】

マルチスレッド実行方法の概要を説明する図である。

【図１６】

従来のプログラム並列化装置の構成例を示すブロック図である。

【符号の説明】

１３　逐次処理プログラム

１４　並列化プログラム

１５　入力データ

２０，２０’，２０”　プログラム並列化装置

２１　制御・データフロー解析部

２２　フォーク箇所候補決定部

２３，２３’，２３”　フォーク箇所候補最良組合せ決定部

２４　並列実行性能評価部

２５　並列化プログラム出力部

１００，１２０　フォーク箇所候補生成ステップ

１０１，１２１　フォーク箇所候補順序付けステップ

１０２，１２２　フォーク箇所候補初期組合せ生成ステップ

１０３，１２３　逐次実行トレース生成ステップ

１０４，１２４　スレッド要素分割ステップ

１０５，１２５　スレッド要素情報解析・記憶ステップ

１０６，１２６　フォーク箇所候補選択ステップ

１０７，１２７　フォーク箇所候補最良組合せ判定ステップ

１０８，１２８　フォーク箇所候補加入・試行組合せ設定ステップ

１０９，１２９　フォーク箇所候補削除・試行組合せ設定ステップ

１１０，１３０　並列実行性能評価ステップ

１１１，１３１　並列実行性能比較ステップ

１１２，１３２　最良組合せ設定ステップ

１１３，１３３　全フォーク箇所候補選択判定ステップ

１１４，１３４　新最良組合せ発見判定ステップ

１１５，１３５　全フォーク箇所候補未選択設定ステップ

１１６，１３６　最良組合せ出力ステップ

１３７　キャンセルフォーク箇所候補削除ステップ

１３８　キャンセルフォーク箇所候補削除ステップ

１４０　グループ分割処理呼び出しステップ

１４１　グループに対するフォーク箇所候補最良組合せ決定処理ステップ

１４２　全グループ処理判定ステップ

１４３　結果出力ステップ

１４６　フォーク箇所候補数・分割下限数比較ステップ

１４７　フォーク箇所候補分割ステップ

１４８　集合のグループへの分割ステップ

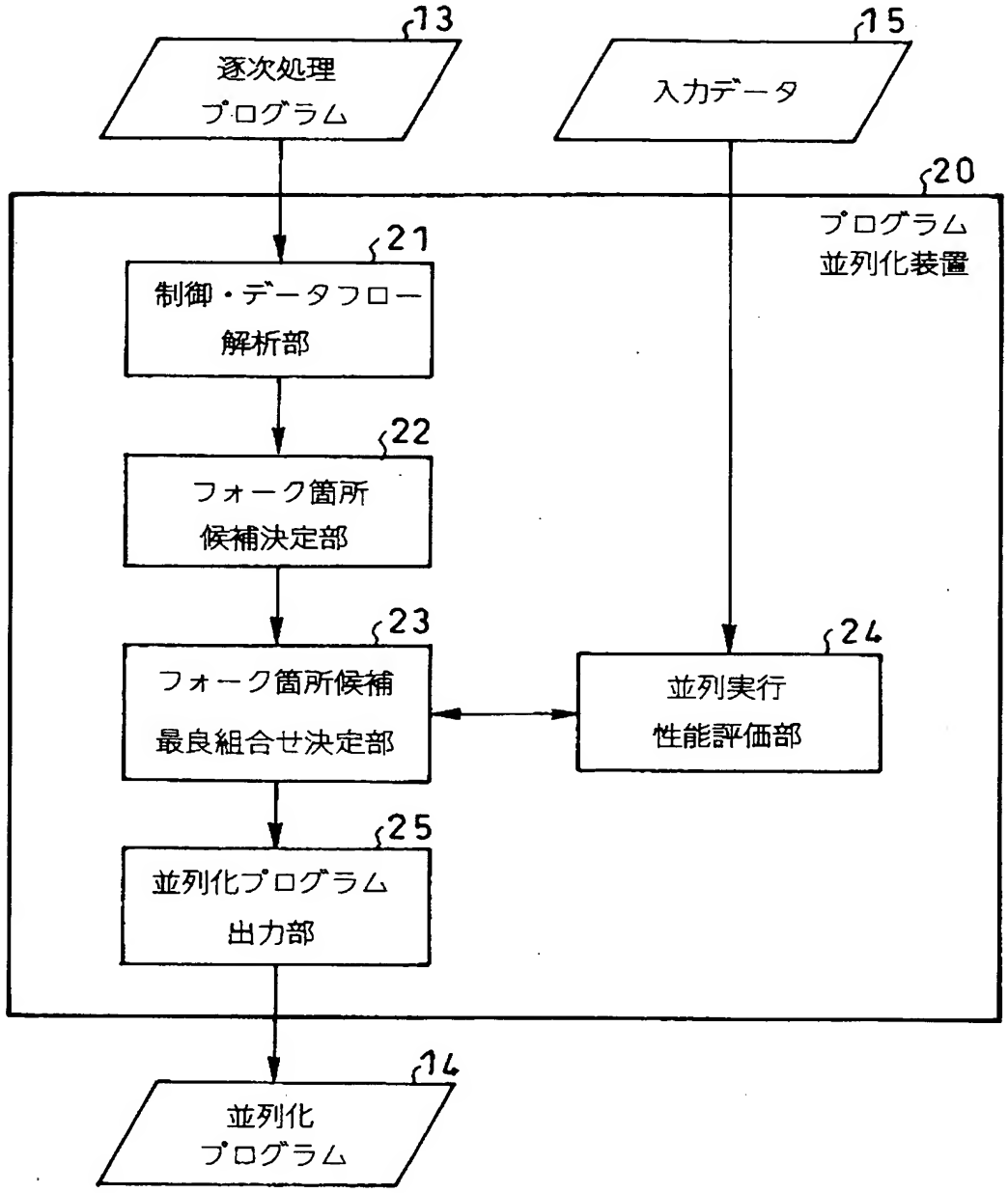
１４９　グループ分割処理呼び出しステップ

１５０　グループ分割処理呼び出しステップ

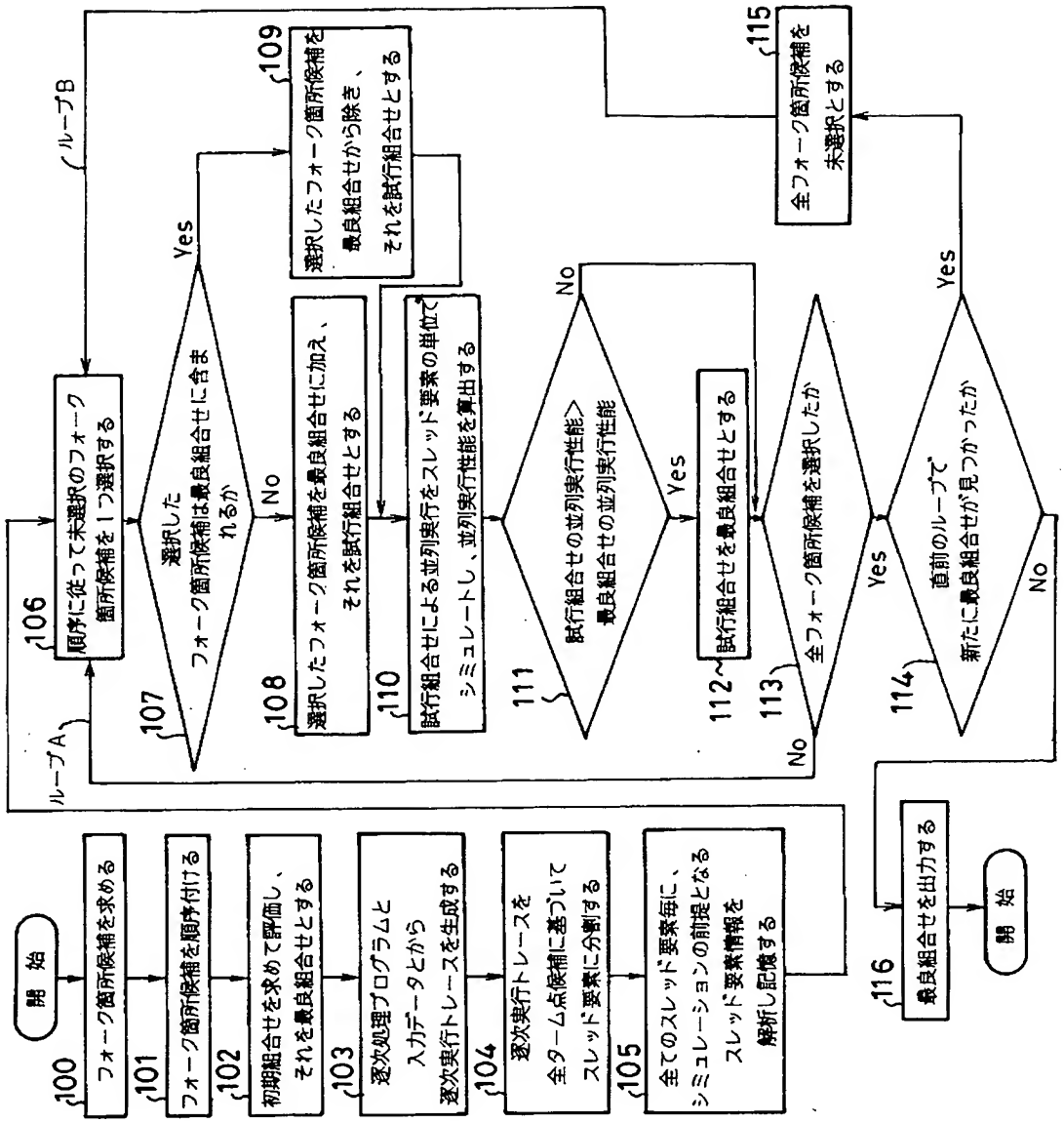
４００，４００’，４００”　プログラム並列化プログラム

【書類名】　図面

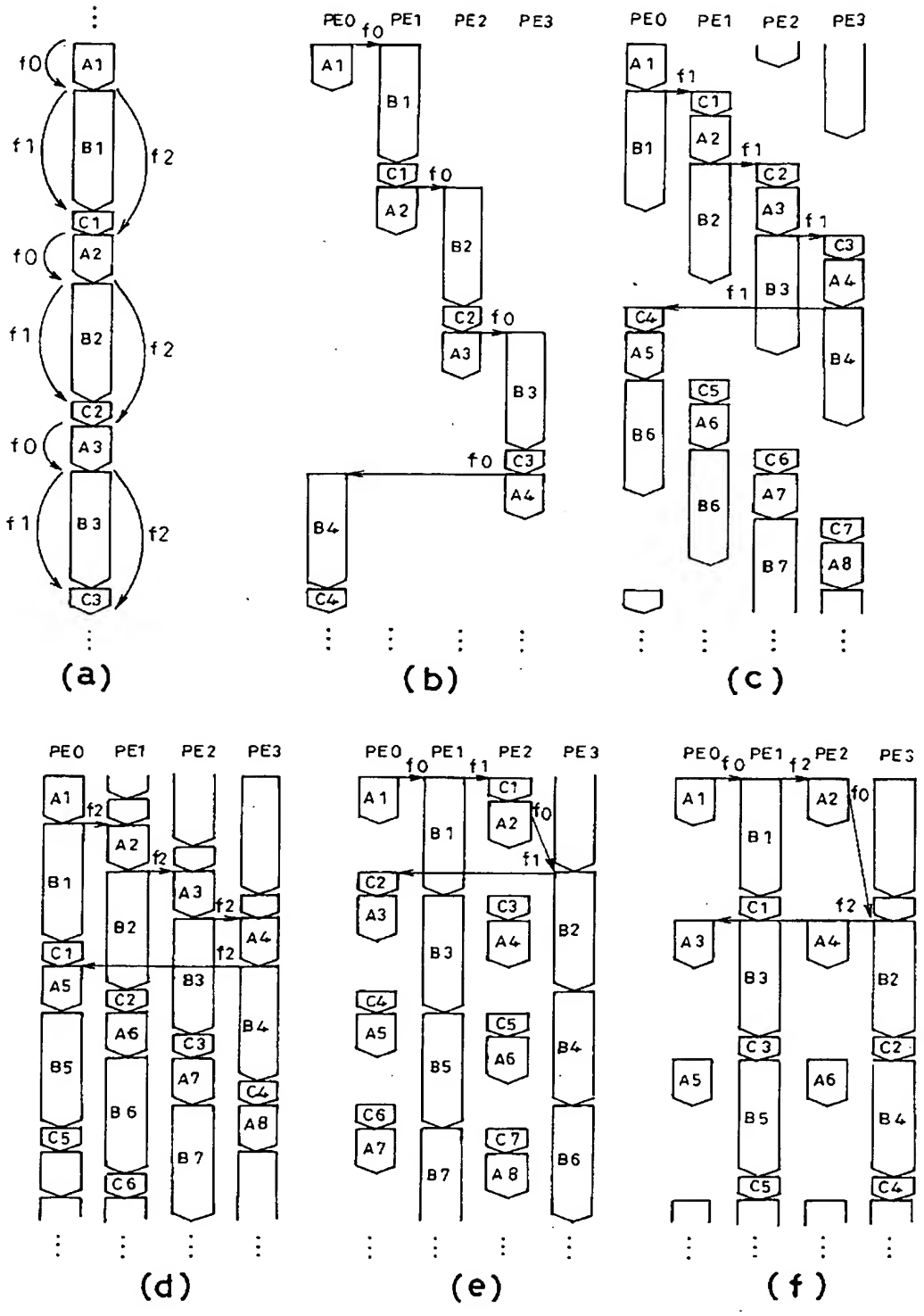
【図１】


13
逐次処理
プログラム
15
入力データ
20
プログラム
並列化装置
21
制御・データフロー
解析部
22
フォーク箇所
候補決定部
23
フォーク箇所候補
最良組合せ決定部
24
並列実行
性能評価部
25
並列化プログラム
出力部
14
並列化
プログラム

【図２】


開始
100 フォーク箇所候補を求める
101 フォーク箇所候補を順序付ける
102 初期組合せを求めて評価し、それを最良組合せとする
103 逐次処理プログラムと入力データとから逐次実行トレースを生成する
104 逐次実行トレースを全タイム点候補に基づいてスレッド要素に分割する
105 全てのスレッド要素毎に、シミュレーションの前提となるスレッド要素情報を解析し記憶する
106 順序に従って未選択のフォーク箇所候補を１つ選択する
107 選択したフォーク箇所候補は最良組合せに含まれるか
Yes
No
108 選択したフォーク箇所候補を最良組合せに加え、それを試行組合せとする
109 選択したフォーク箇所候補を最良組合せから除き、それを試行組合せとする
110 試行組合せによる並列実行をスレッド要素の単位でシミュレートし、並列実行性能を算出する
111 試行組合せの並列実行性能＞最良組合せの並列実行性能
112 試行組合せを最良組合せとする
113 全フォーク箇所候補を選択したか
114 直前のループで新たに最良組合せが見つかったか
115 全フォーク箇所候補を未選択とする
116 最良組合せを出力する
終了
ループA
ループB

【図３】


PE0
PE1
PE2
PE3
f0
f1
f2
A1
B1
C1
A2
B2
C2
A3
B3
C3
A4
B4
C4
A5
B5
C5
A6
B6
C6
A7
B7
C7
A8
(a)
(b)
(c)
(d)
(e)
(f)

【図４】

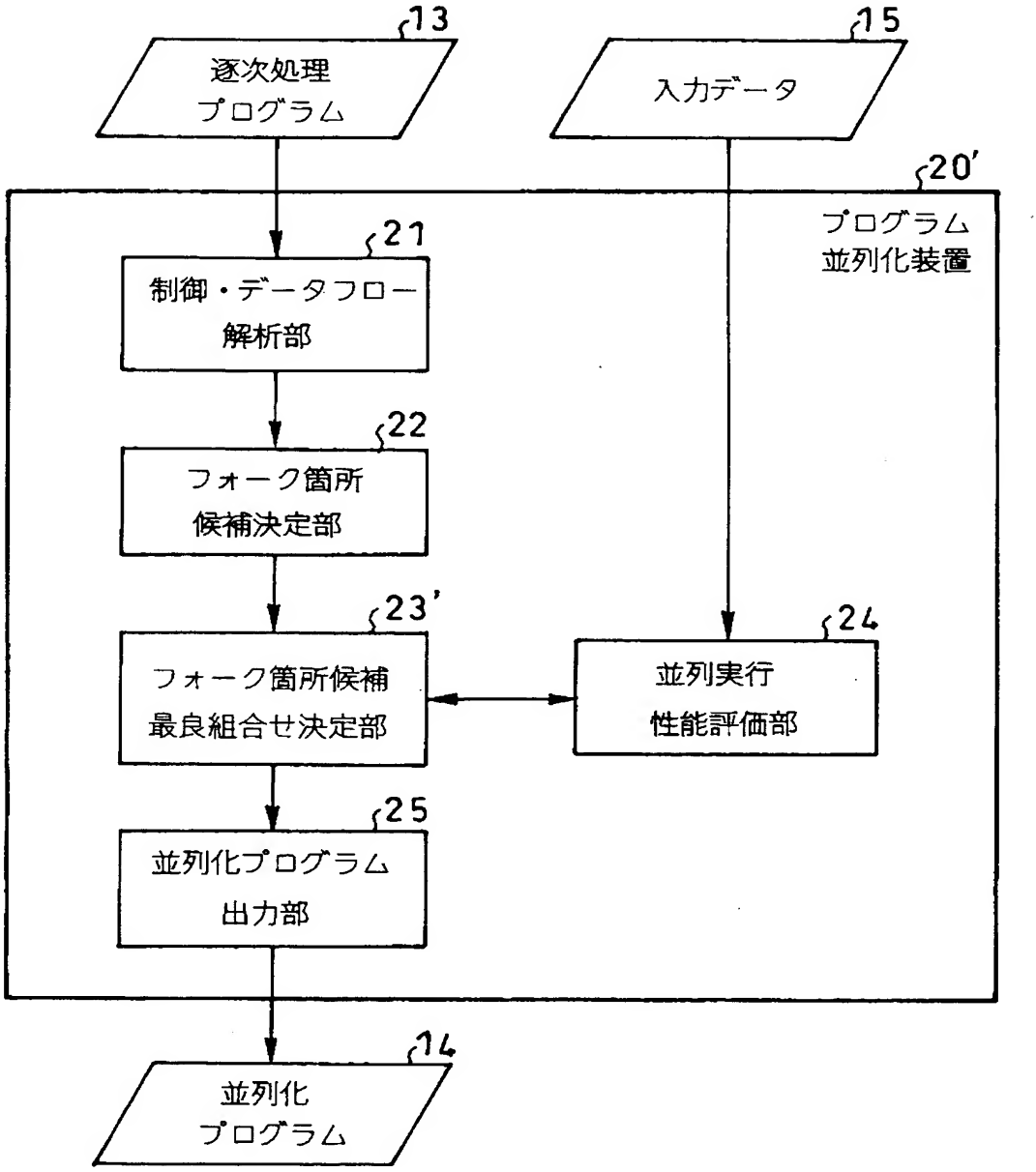
フォーク箇所候補組合せ	並列実行性能(倍)
φ	1.0
{f0}	2.5
{f1}	2.2
{f2}	2.4
{f3}	1.8
{f0,f1}	3.0
{f0,f2}	3.2
{f0,f3}	2.4
{f1,f2}	3.8
{f1,f3}	2.8
{f2,f3}	2.0
{f0,f1,f2}	3.5
{f0,f1,f3}	2.8
{f0,f2,f3}	2.5
{f1,f2,f3}	2.6
{f0,f1,f2,f3}	2.5

【図５】

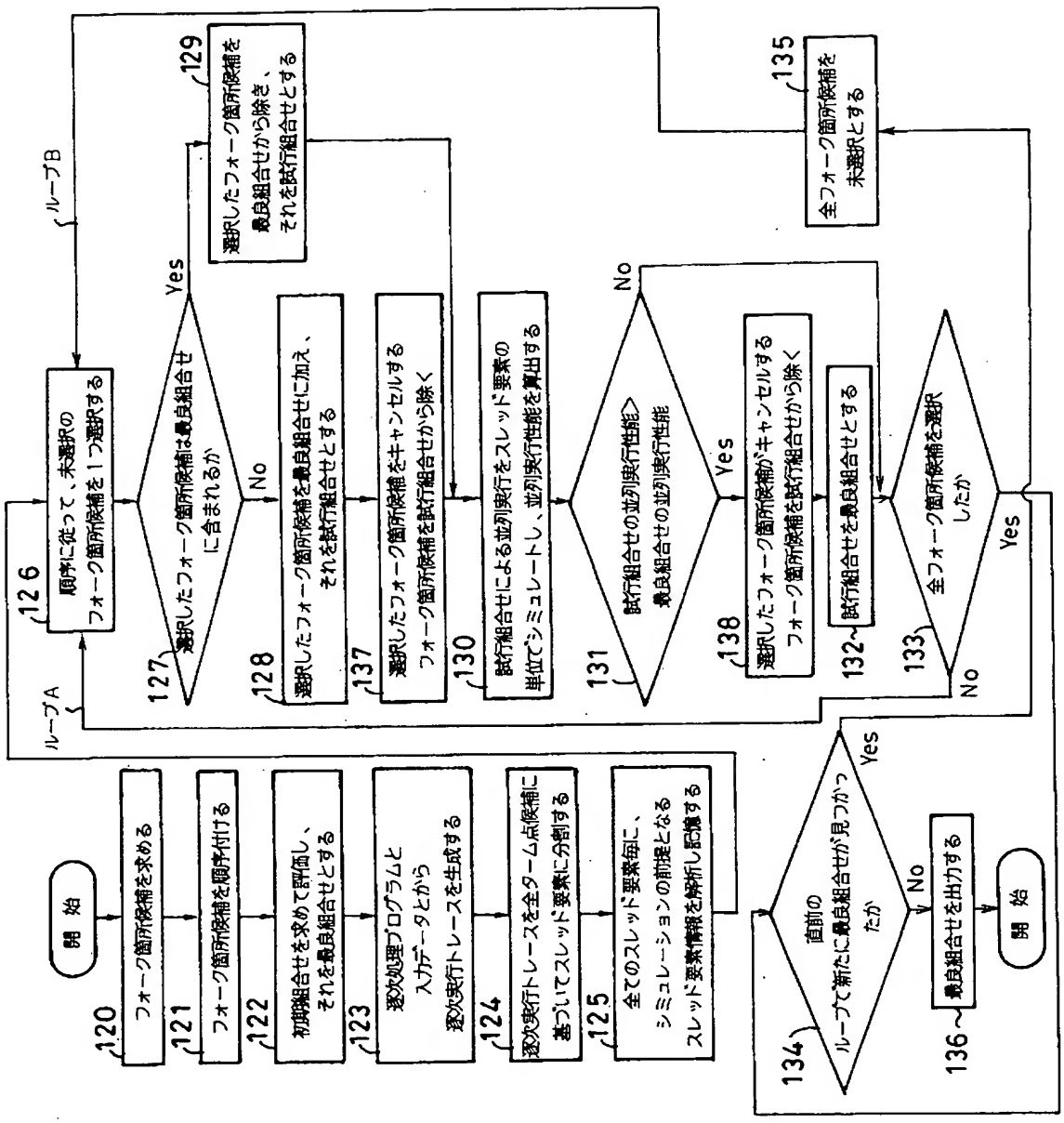
初期組合せ：｛ｆ０，ｆ２｝　（並列実行性能：３．２倍）

	選択フォーク箇所候補	並列実行性能		
	試行組合せ		最良組合せ	選択済フォーク箇所候補
繰り返し１－１	f0 {f2}	2.4	{f0,f2}	f0
繰り返し１－２	f2 {f0}	2.5	{f0,f2}	f0,f2
繰り返し１－３	f1 {f0,f1,f2}	3.5	{f0,f1,f2}	f0,f1,f2
繰り返し１－４	f3 {f0,f1,f2,f3}	2.5	{f0,f1,f2}	f0,f1,f2,f3
繰り返し２－１	f0 {f1,f2}	3.8	{f0,f1,f2}	f0
繰り返し２－２	f2 {f1}	2.2	{f1,f2}	f0,f2
繰り返し２－３	f1 {f2}	2.4	{f1,f2}	f0,f1,f2
繰り返し２－４	f3 {f1,f2,f3}	2.6	{f1,f2}	f0,f1,f2,f3
繰り返し３－１	f0 {f0,f1,f2}	3.5	{f1,f2}	f0
繰り返し３－２	f2 {f1}	2.2	{f1,f2}	f0,f2
繰り返し３－３	f1 {f2}	2.4	{f1,f2}	f0,f1,f2
繰り返し３－４	f3 {f1,f2,f3}	2.6	{f1,f2}	f0,f1,f2,f3

【図６】


13
逐次処理
プログラム
15
入力データ
20'
プログラム
並列化装置
21
制御・データフロー
解析部
22
フォーク箇所
候補決定部
23'
フォーク箇所候補
最良組合せ決定部
24
並列実行
性能評価部
25
並列化プログラム
出力部
14
並列化
プログラム

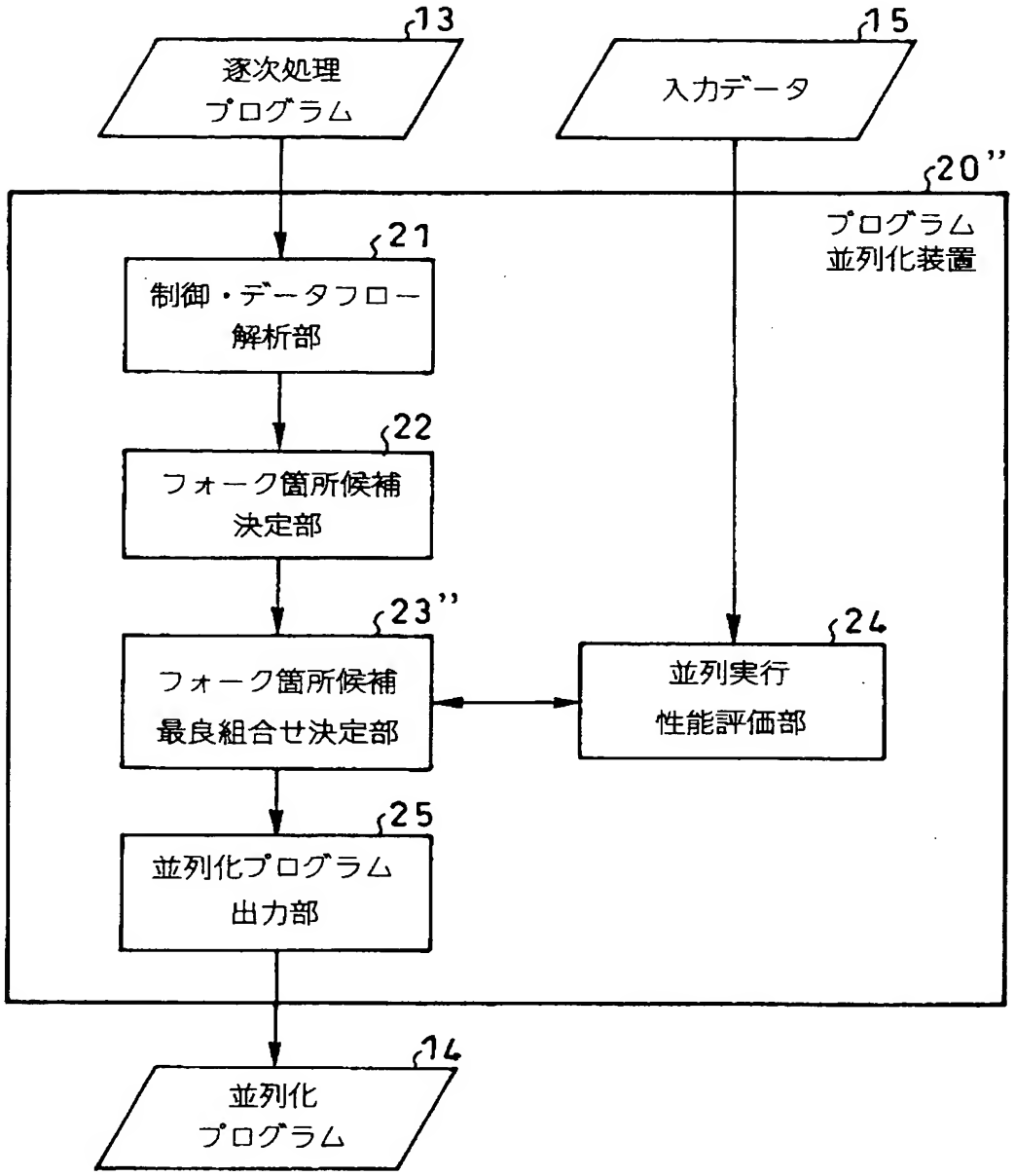
【図７】


開 始
120 フォーク箇所候補を求める
121 フォーク箇所候補を順序付ける
122 初期組合せを求めて評価し、それを最良組合せとする
123 逐次処理プログラムと入力データとから逐次実行トレースを生成する
124 逐次実行トレースを全タイム点候補に基づいてスレッド要素に分割する
125 全てのスレッド要素毎に、シミュレーションの前提となるスレッド要素情報を解析し記憶する
134 ループで新たに最良組合せが見つかったか
Yes
No
136 最良組合せを出力する
開 始
ループA
126 順序に従って、未選択のフォーク箇所候補を１つ選択する
127 選択したフォーク箇所候補は最良組合せに含まれるか
Yes
No
ループB
129 選択したフォーク箇所候補を最良組合せから除き、それを試行組合せとする
128 選択したフォーク箇所候補を最良組合せに加え、それを試行組合せとする
137 選択したフォーク箇所候補をキャンセルするフォーク箇所候補を試行組合せから除く
130 試行組合せによる並列実行をスレッド要素の単位でシミュレートし、並列実行性能を算出する
131 試行組合せの並列実行性能＜最良組合せの並列実行性能
No
Yes
138 選択したフォーク箇所候補がキャンセルするフォーク箇所候補を試行組合せから除く
132 試行組合せを最良組合せとする
133 全フォーク箇所候補を選択したか
No
Yes
135 全フォーク箇所候補を未選択とする

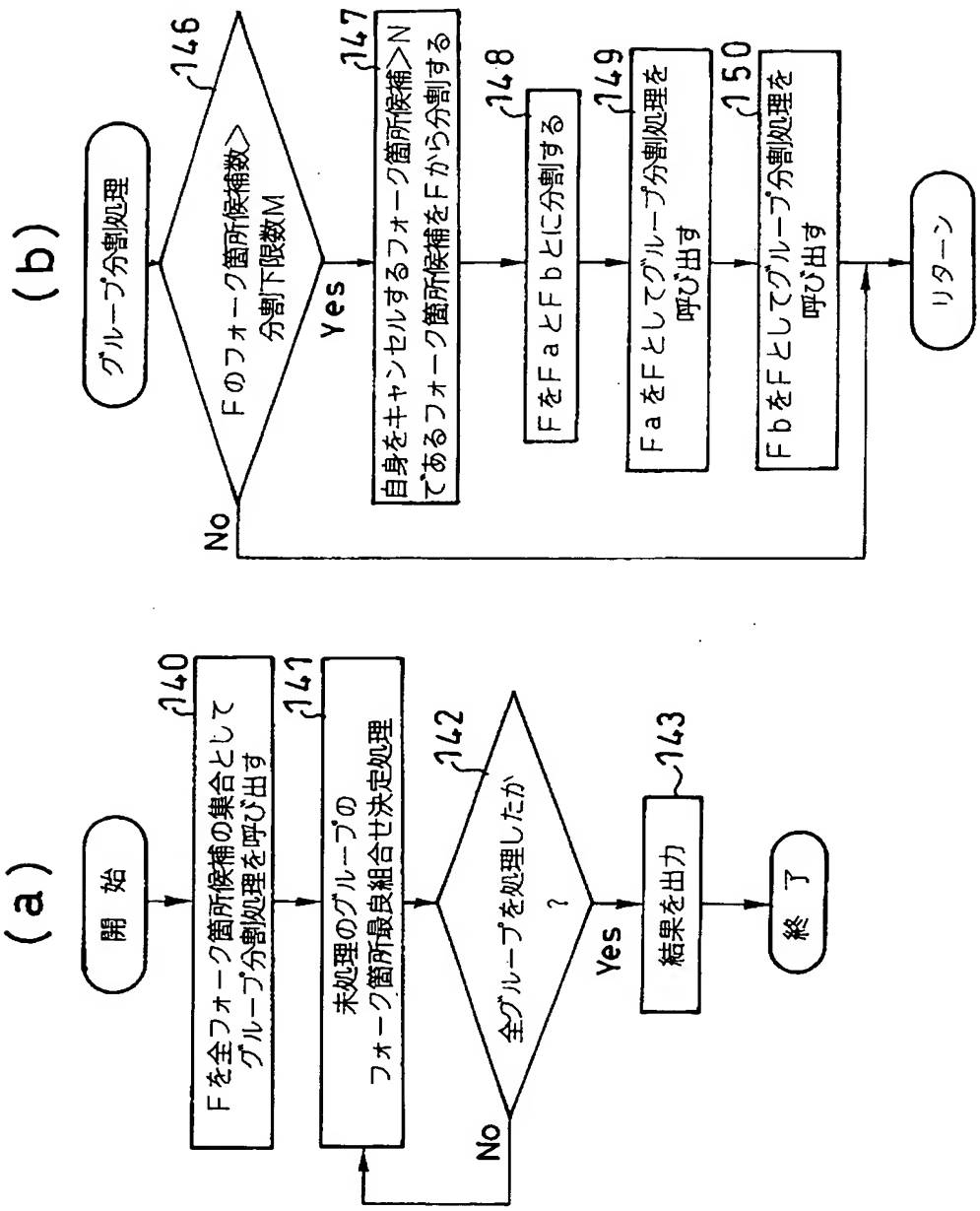
初期組合せ：｛ｆ０，ｆ２｝　（並列実行性能：３．２倍）

	選択フォーク箇所候補	並列実行性能		
	試行組合せ		最良組合せ	選択済フォーク箇所候補
繰り返し１−１	f0 {f2}	2.4	{f0,f2}	f0
繰り返し１−２	f2 {f0}	2.5	{f0,f2}	f0,f2
繰り返し１−３	f1 {f1,f2}	3.8	{f1,f2}	f0,f1,f2
繰り返し１−４	f3 {f1,f3}	2.8	{f1,f2}	f0,f1,f2,f3
繰り返し２−１	f0 {f0,f1,f2}	3.5	{f1,f2}	f0
繰り返し２−２	f2 {f1}	2.2	{f1,f2}	f0,f2
繰り返し２−３	f1 {f2}	2.4	{f1,f2}	f0,f1,f2
繰り返し２−４	f3 {f1,f3}	2.8	{f1,f2}	f0,f1,f2,f3

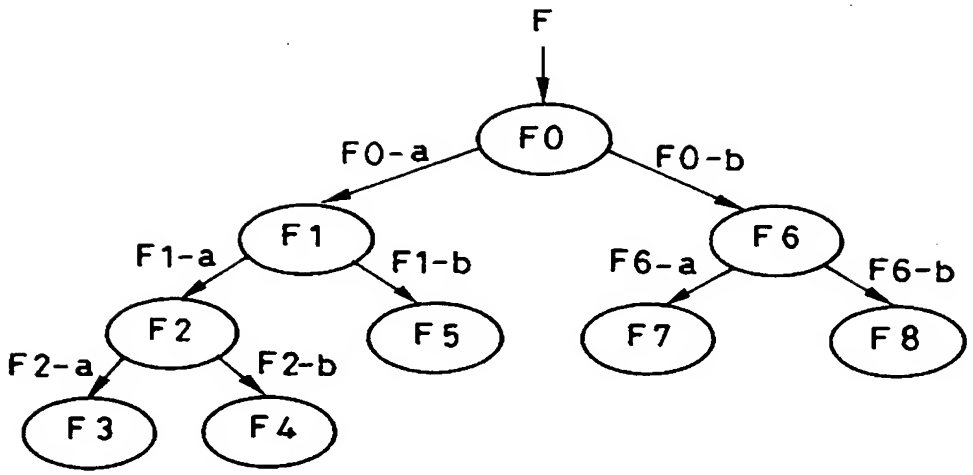
【図９】


13
逐次処理
プログラム
15
入力データ
20''
プログラム
並列化装置
21
制御・データフロー
解析部
22
フォーク箇所候補
決定部
23''
フォーク箇所候補
最良組合せ決定部
24
並列実行
性能評価部
25
並列化プログラム
出力部
14
並列化
プログラム

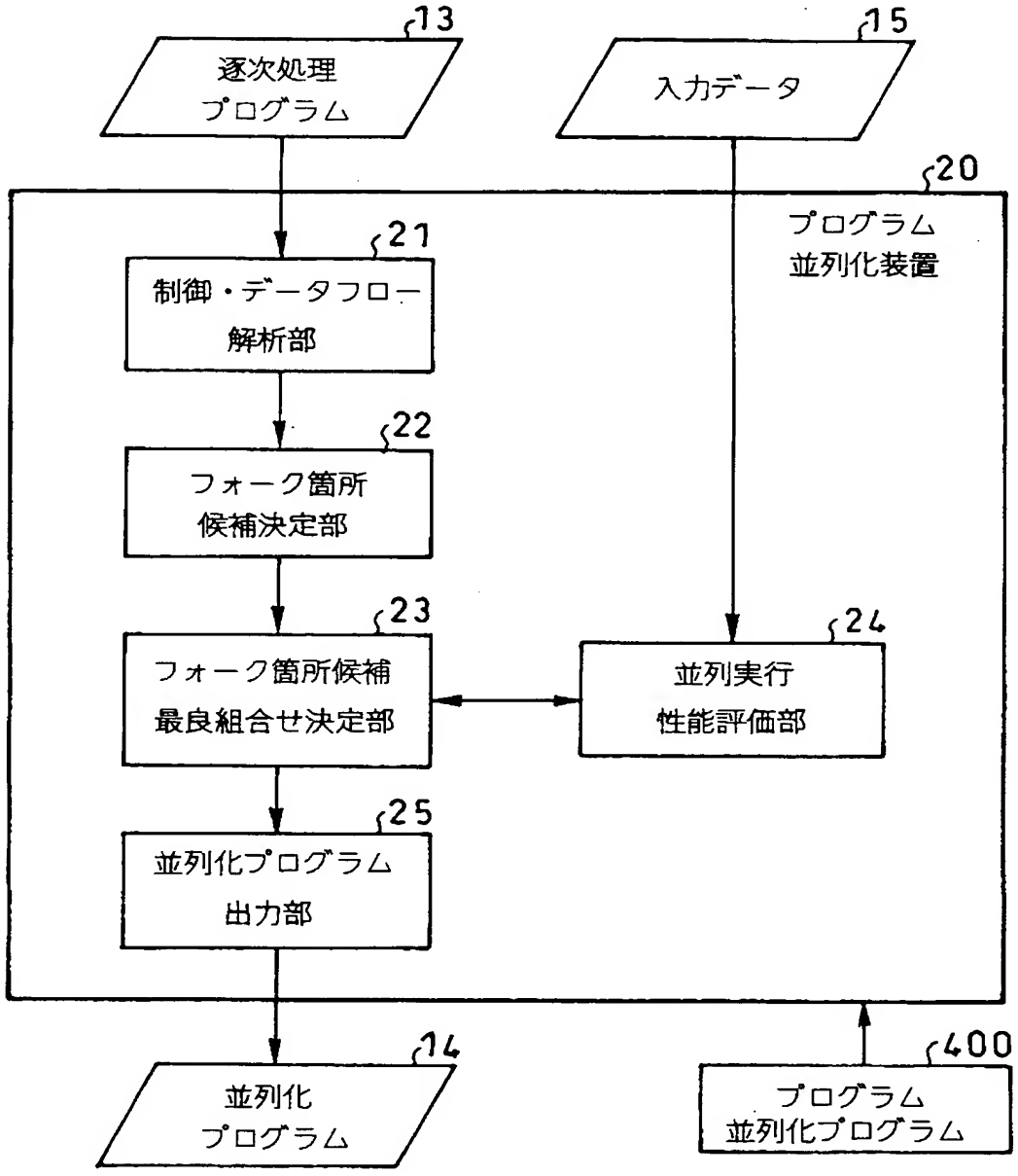
【図１０】


(a)
開　始
140
Ｆを全フォーク箇所候補の集合として
グループ分割処理を呼び出す
141
未処理のグループの
フォーク箇所最良組合せ決定処理
142
全グループを処理したか
？
No
Yes
143
結果を出力
終　了
(b)
グループ分割処理
146
Ｆのフォーク箇所候補数＞
分割下限数Ｍ
No
Yes
147
自身をキャンセルするフォーク箇所候補＞Ｎ
であるフォーク箇所候補をＦから分割する
148
ＦをＦａとＦｂとに分割する
149
Ｆａを Ｆとしてグループ分割処理を
呼び出す
150
Ｆｂを Ｆとしてグループ分割処理を
呼び出す
リターン

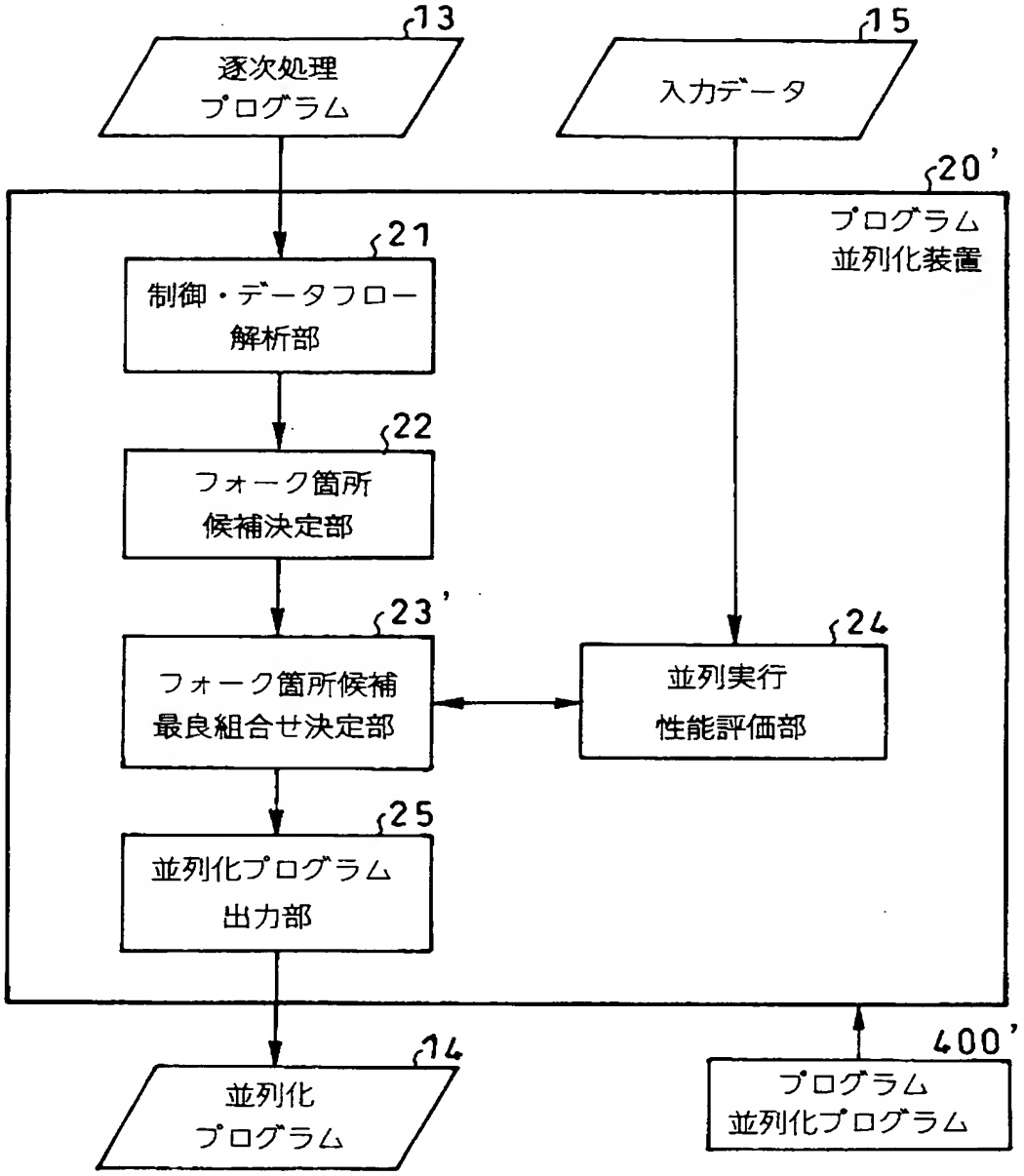
【図１１】



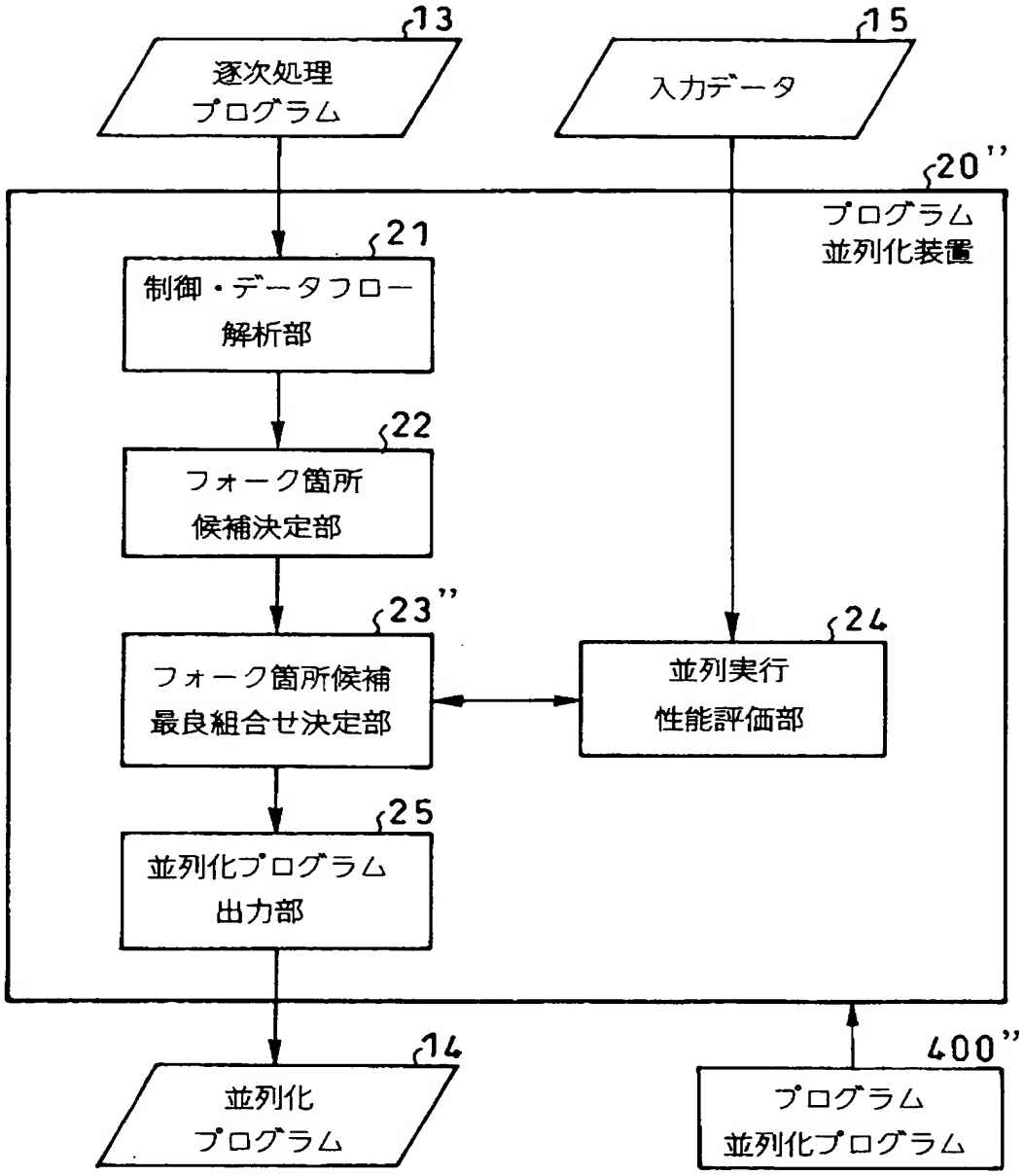
【図１２】


13
逐次処理
プログラム
15
入力データ
20
プログラム
並列化装置
21
制御・データフロー
解析部
22
フォーク箇所
候補決定部
23
フォーク箇所候補
最良組合せ決定部
24
並列実行
性能評価部
25
並列化プログラム
出力部
14
並列化
プログラム
400
プログラム
並列化プログラム

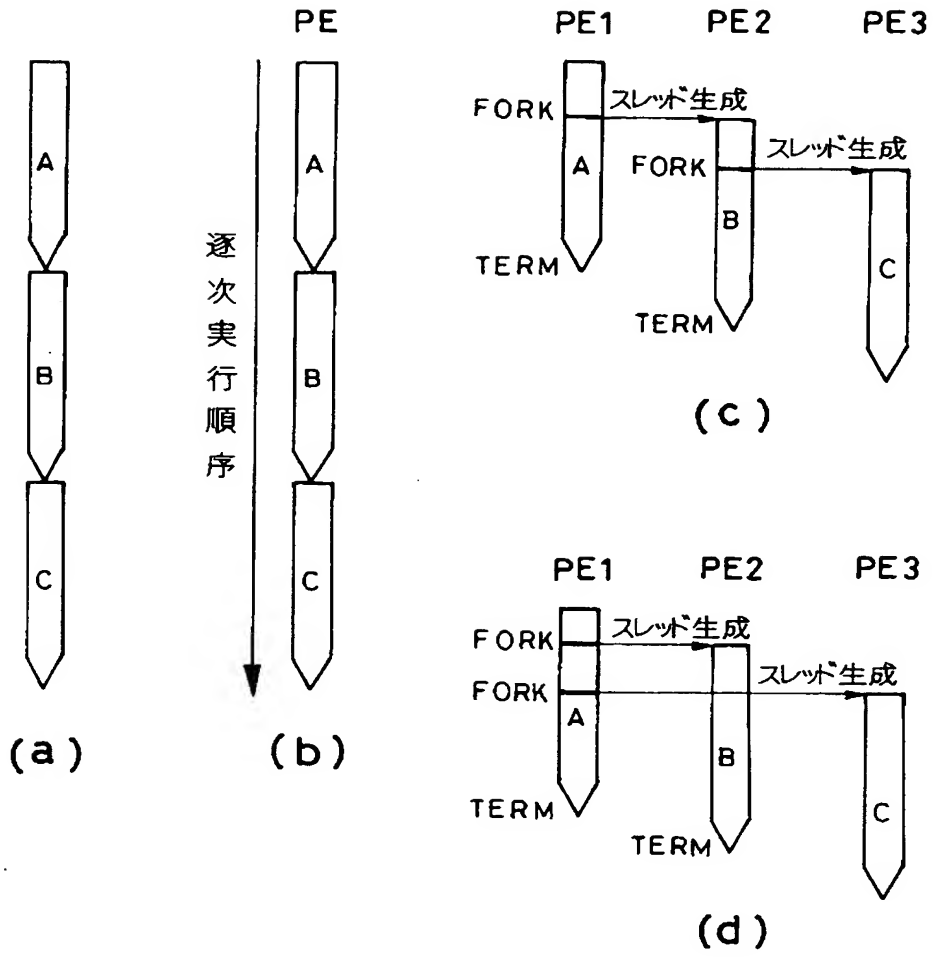
【図１３】


13
逐次処理
プログラム
15
入力データ
20'
プログラム
並列化装置
21
制御・データフロー
解析部
22
フォーク箇所
候補決定部
23'
フォーク箇所候補
最良組合せ決定部
24
並列実行
性能評価部
25
並列化プログラム
出力部
14
並列化
プログラム
400'
プログラム
並列化プログラム

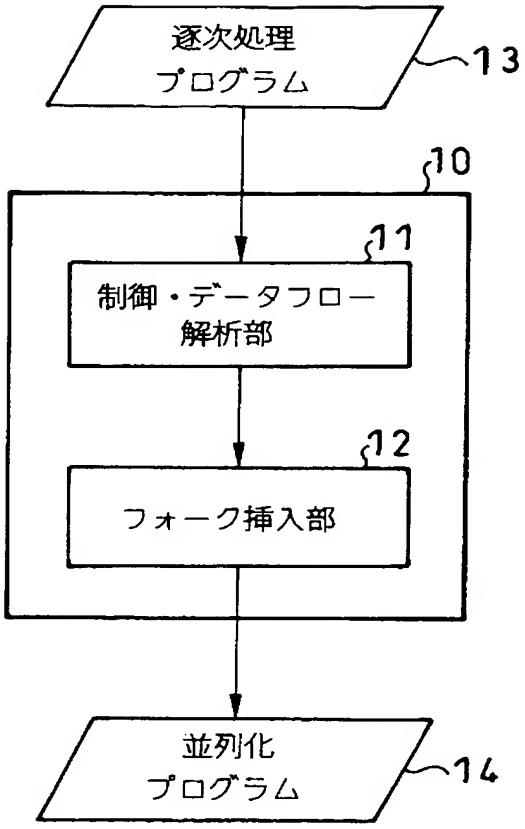
【図１４】


13
逐次処理
プログラム
15
入力データ
20''
プログラム
並列化装置
21
制御・データフロー
解析部
22
フォーク箇所
候補決定部
23''
フォーク箇所候補
最良組合せ決定部
24
並列実行
性能評価部
25
並列化プログラム
出力部
14
並列化
プログラム
400''
プログラム
並列化プログラム

【図１５】


PE
A
B
C
逐次実行順序
(a)
(b)
PE1
PE2
PE3
FORK
スレッド生成
A
FORK
スレッド生成
B
TERM
C
TERM
(c)
PE1
PE2
PE3
FORK
スレッド生成
FORK
スレッド生成
A
B
TERM
C
TERM
(d)

【図１６】



【書類名】　要約書

【要約】

【課題】逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法向けに、並列実行性能のより高い並列化プログラムを高速に生成する。

【解決手段】制御・データフロー解析部２１は、逐次処理プログラム１３の制御フローおよびデータフローの解析を行い、それを参照してフォーク箇所候補決定部２２がフォーク箇所候補を決定する。フォーク箇所候補最良組合せ決定部２３は、並列実行性能評価部２４によってフォーク箇所候補の試行組合せの並列実行性能を評価した結果を基準にフォーク箇所候補の最良組合せを決定し、並列化プログラム出力部２５はフォーク箇所候補の最良組合せに基づいてフォーク命令を挿入して並列化プログラム１４を生成し出力する。

【選択図】図１

認定・付加情報

特許出願の番号	特願２００３－０９３０７６
受付番号	５０３００５２２９４５
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１５年　４月　１日

＜認定情報・付加情報＞

【提出日】　平成15年 3月31日

次頁無

特願２００３－０９３０７６

出　願　人　履　歴　情　報

識別番号　　［０００００４２３７］

１．変更年月日　　１９９０年　８月２９日
［変更理由］　　新規登録
住　所　　東京都港区芝五丁目７番１号
氏　名　　日本電気株式会社